オールMSX BASIC 用語・用例新辞典

秋本京子著

成美堂出版

この本はMSX·BASICの命令語を①ダイレクトコマンド②画面表示コマンド③一般ステートメント④グラフィックコマンド⑤音楽・音・ブザー⑥割り込み処理⑦関数⑧ディスクコマンド⑨ファイル制御の9つのパートに分け、それぞれに、その語の定義、書式、説明、書式例、サンプルプログラムを示したものです。また、MSXの基礎知識、MSX-DOSについても詳述してあります。
パソコン操作で行き詰まったとき、また、マニュアルや他の手引き書を見ても疑問が解けないとき、この本の各項目をひいてください。必ず解答が得られるはずです。

オールMSX BASIC 用語・用例新辞典

秋本京子著

成美堂出版

＊序に代えて

ＭＳＸと本書の利用方法

ＭＳＸタイプのコンピュータはたしかに画期的なパーソナルコンピュータです。コンピュータ自体のもつ機能のすばらしさもそうですが、それよりもなによりも、ＭＳＸタイプのコンピュータなら、どの機種も共通したＢＡＳＩＣで動くというところに、利用範囲の広がりが感じられるのです。

在来の他機種のことを考えてみてください。そのマシンの潜在的な機能がどれほどすぐれたものであっても、それを引き出すためのパッケージソフトに不足していたというのが実情でしょう。なぜなら、たとえ、他の機種用に開発したすぐれたパッケージプログラムがそこにあったとしても、コンピュータ言語のＢＡＳＩＣが通じ合わないために、自分の機械でそれを利用できなかったからです。

その点ＭＳＸタイプのコンピュータは世界が違います。どのＭＳＸタイプのコンピュータのために作られたプログラムであっても、ＭＳＸタイプのコンピュータでさえあれば、自由自在に利用できます。したがって、これから市場に現れるパッケージソフトは、他の機種のソフトにくらべてどんどんふえていくはずです。

ＭＳＸのすばらしさは、その共通ＢＡＳＩＣにあるといっていいのです。誇ってもいいＢＡＳＩＣの共通システムなのです。

これで、ハードウェアよし、ソフトウェアよしとおぜんだてはととのったわけですが、それを実のあるものにするか、虚のままに終わらせるかは、ひとえにこのすぐれたＢＡＳＩＣをどう使いこなすかにかかっています。

人間のことばもコンピュータのことばも、最初は無条件に暗記することから始まります。ことばとは暗記科目なのです。ことばを暗記しなければ会話はで

きません。この本を手にしたら、項目ごとに各命令語の意味を暗記してください。いままでに、だいたい憶えていた人でも、少し違った思い込みかたをしていたことに気付く人が出てくるでしょう。

ＢＡＳＩＣコマンドは、ひとつの意味を表現しているだけではなく、使い方しだいではいろいろな意味があります。そのときどきの条件の違いに応じた使い方にも目を向けてください。ひとつのコマンドだけではなく、他のコマンドとの関係にも注視して、ようやく１コマンド暗記の部を卒業というわけです。

そして、なによりも重要なのが、暗記したＢＡＳＩＣコマンドをさまざまに組み立てて、自分のやりたいことをコンピュータにやらせるためのプログラムを作りあげていくことです。本書では、各項目の末尾にスペースの許すかぎり、例としてのサンプルプログラムを組み込んであります。このプログラムが、なにをどうしようとしているのか、ひと目で読みとれるようになったとき、一人前のマイコニストとしての出発点に立ったといっていいでしょう。サンプルプログラムを入力して実行させながら、プログラムの骨子を自分のものにしていってください。

著者しるす

目　次

コンピュータ操作の前に

MSX-DOS入門

MSX BASIC用語用例辞典

❶ダイレクトコマンド

②画面表示コマンド

③一般ステートメント

❹グラフィックコマンド

❺音楽・音・ブザー

❻割り込み処理

⑦関　数

⑧ディスクコマンド

⑨ファイル制御

索　引

A

B

C

D

R

S

T

V

W

コンピュータ操作の前に

❶コンピュータ使用上の注意

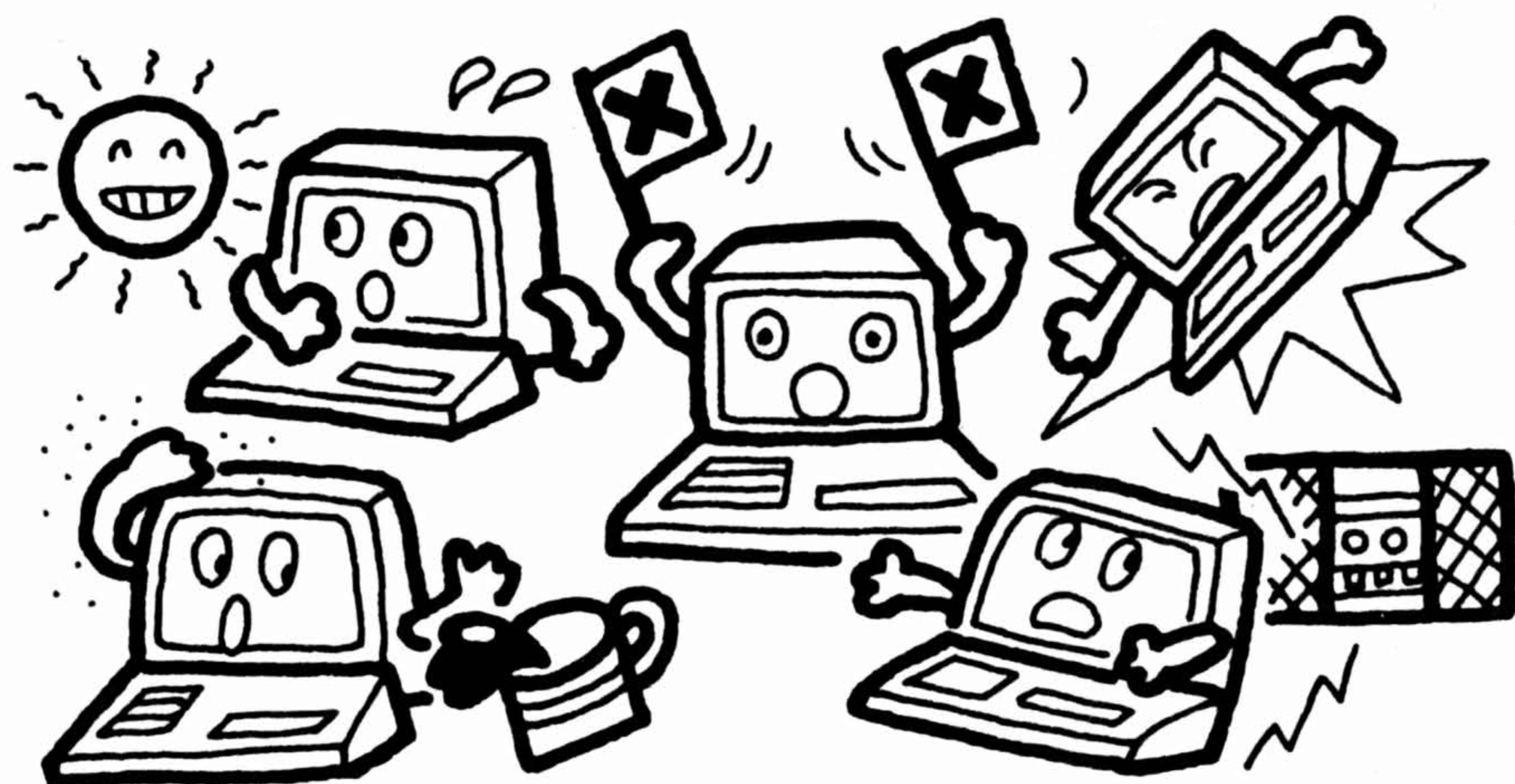
コンピュータは精密機械である。操作の前に、まずその取り扱いにじゅうぶん注意を払うことが、機械を正確に作動させ、長持ちさせる。ひいては一人前のマイコニストになるための前提条件である。

(1)　高温、多湿、急激な温度変化は避ける

長時間直射日光に当たる場所、暖房器の近くなど、高温になりやすい場所には絶対に置かないこと。また、湿気の多い場所や、低温から高温へ内部温度が急激に変化するような温度差の激しいところで使用すると、露が付着して異常が発生することがある。

(2)　ホコリ、水分、汚れに注意

ホコリの多い場所を避け、操作しない時はカバーをかけておく習慣をつける。カバーがなければふろしきやタオルケットでもかまわない。また、コーヒーやジュースなどの飲み物をこぼしたり、油分のついた汚れた手で触れない。

パソコンの外箱やモニタ・テレビの画面、キーボードの部分などの汚れは乾いた布で軽く拭き取る。水や洗剤、揮発性液体(シンナー、ベンジンなど)は使用しないこと。

(3)　衝撃、振動を与えない

パソコンは精密な電子部品からできており、プリント基盤をもとにしたパターン配線がされているため、落としたりぶつけたり、強いショックを与えると内部破損の原因になる。振動にも弱いため、常時小さな振動のあるような場所は避ける。

(4)　電子機器などの近くに置かない

電子機器などの近くに置いて使用すると、電磁波の影響でノイズ（雑音）が入ったり、モニタの画面がぶれたり色が乱れたりすることがある。ラジオやテレビ、オーディオ・チューナの近くに設置しないこと。

❷コンピュータの基本機能

コンピュータは、次の５つの基本機能から構成されている。

1.入力機能――外部から情報を取り入れる機能。パソコンではキーボードが入力装置（input unit）にあたる。

2.記憶機能――入力装置から入力されたデータやプログラムを記憶（格納）する機能。

この機能を果たす記憶装置（memory unit）には、パソコン内部にデータやプログラムを記憶する主記憶装置と、パソコン外部に保存するための補助記憶装置がある。パソコンではカセットレコーダやフロッピーディスクが補助記憶装置にあたる。

また、この記憶装置には大きく分けて２種類ある。

●読み出し専用のＲＯＭ（ロム＝Read Only Memory）

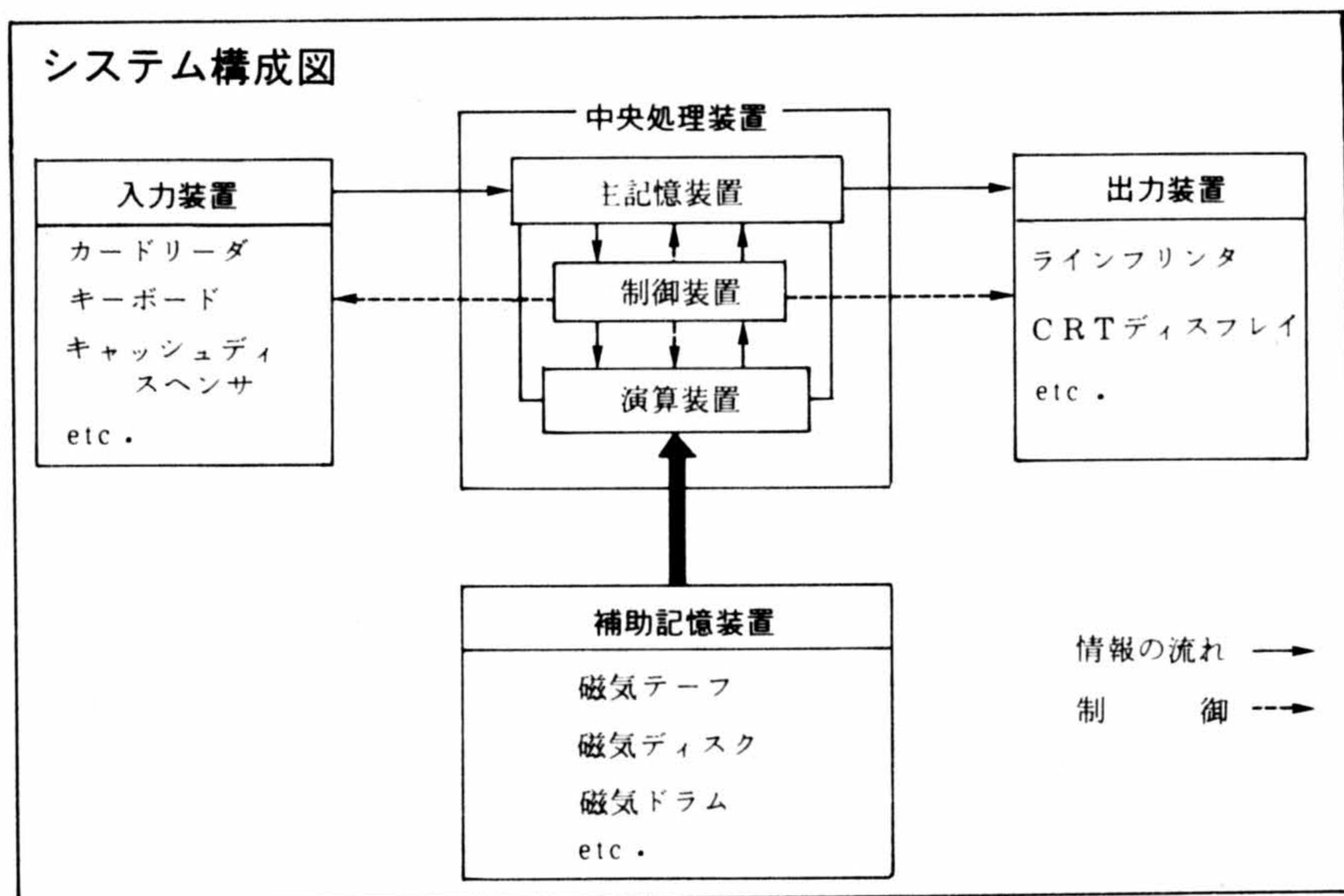
あらかじめメモリに記憶されているデータを読み出して再生するメモリ。新しいデータを記録することはできない。ＭＳＸパソコンでは、ＭＳＸ・ＢＡＳＩＣが本体に内臓されており（メーカーで記憶済み）、ＭＳＸ・ＢＡＳＩＣで書かれているプログラムの解釈や実行がなされる。

●読み出し書き込み自由のＲＡＭ（ラム＝Random Access Memory）

読み出し専用のＲＯＭにくらべてアクセスが自由なＩＣメモリ。キーボードからデータやプログラムを自由に入力し、またそれを自由に読み出すことができる。

3.演算機能——コンピュータが記憶された情報をもとにいろいろな計算や判断、組み合わせなどをする機能。パソコン内部の演算装置（arithmetic unit）にあたり、ここで処理された結果は再び主記憶装置に貯えられる。

4.制御機能——コンピュータの動きをコントロールし、プログラムどおりに正しく作動させる機能。パソコン内部の制御装置（control unit）にあたり、指令室のような役割をする。

5.出力機能——コンピュータ内で処理されたデータをコンピュータ外部へ表示（または書き出す）機能。パソコンではプリンタやディスプレイが出力装置（output unit）にあたる。

❸ＭＳＸとは

機種が違っても同じソフトが使える

ＭＳＸとは、米国のマイクロソフト社（最大手のソフトウェアハウス）と日本のアスキー社（パソコン関連の出版社）が共同して開発した、ハード、ソフト両面にわたるパソコンの規格である。この規格にそってつくられているのが、いわゆるＭＳＸパソコンで、その最大の特徴は、“機種が違っても同じソフトが使える”ことにある。

周知のとおり、現在、パソコンの多くはメーカーごとに違いがある。また、同じメーカーでも機種が違えば何らかの違いがあり、使用者はさまざまな不便を強いられてきた。

その最も大きな要因は、ソフトの互換性がないことである。つまり、あるソフトはある機種でしか使えない、市販のソフトもある決められた機種にしか適応せず、手持ちの機種に合わなければ、それがいかにすぐれたソフト、またはおもしろいソフトであっても断念するよりほかなく、ソフトの互換性はユーザーの強く希望するところであった。

そして登場したのが、ＭＳＸパソコンである。

パソコン共通言語“ＭＳＸ·ＢＡＳＩＣ”

コンピュータという機械に人間の意志を伝え、動作をさせるためのメディアがコンピュータ言語である。

コンピュータ言語には、ＢＡＳＩＣ、ＣＯＢＯＬ（コボル）、ＦＯＲＴＲＡＮ（フォートラン）、ＡＬＧＯＬ（アルゴル）、ＰＩＰＳ（ピプス）、マシン語（機械語）……などがあるが、なかでもＢＡＳＩＣ（Beginner's All purpose Symbolic Instruction Code＝初心者用汎用命令コード）は、その名のとおり、初心者にとっていちばんわかりやすく、簡単に使える言葉である。

ＢＡＳＩＣで使う単語（命令語）は、ほとんどが英語の単語に似ており、それも中学校レベルの初歩的な英語が主体となっているため、他の言葉にくらべ

て非常に親しみやすい。

ただ、これまではＢＡＳＩＣといっても、コンピュータの各製品の機種ごとに少しずつ違いがあり、ある機種で使えるＢＡＳＩＣは他の機械で通じないという不便さがあった。

この不便さをなくしたのが、ＭＳＸ・ＢＡＳＩＣというパソコン共通言語である。現在、各メーカーから多くのＭＳＸタイプ・コンピュータが出ているが、ＢＡＳＩＣはすべての機械に通用する。ソフトを選ぶ際にも手持ちのパソコンにのるかどうかなどを気にせずに利用できるようになったのである。

機種のラインナップとともに、今後のソフトの拡充がますます期待されている。

④MSXとMSX2

画面表示機能、グラフィック機能を強化

ＭＳＸ－ＢＡＳＩＣ Ver. 2.0を搭載したＭＳＸをＭＳＸ2という。

ＭＳＸ2は、従来のＭＳＸ－ＢＡＳＩＣの機能をすべて継承したうえで、次の点が大幅に拡張されている。

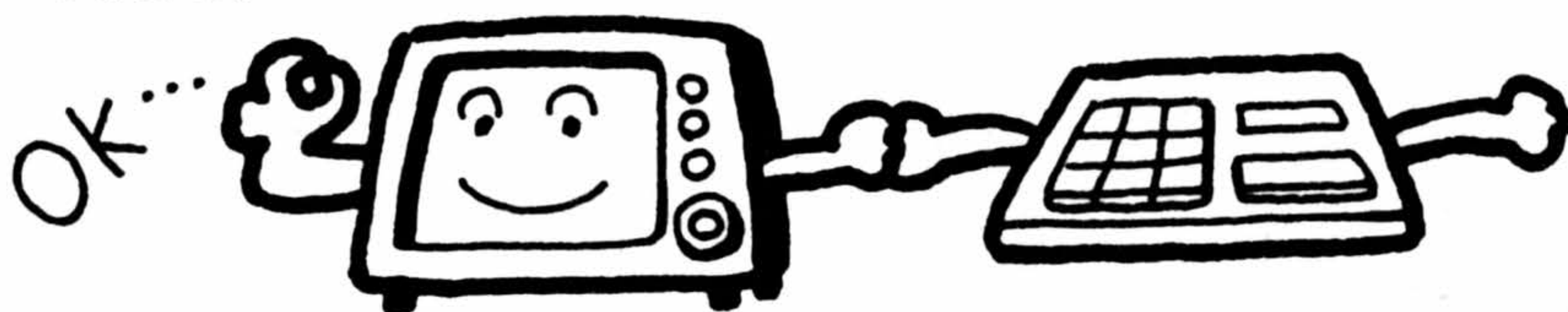
① １行の表示文字数が40文字から80文字へと増加した。

② 複数の画面を持ち、切り換え機能が付いた。

③ グラフィックの解像度が512×212と、より細かな絵や図形を描くことができるようになった。

④ 512色中16色および256色を選択して、ドットごとに色を付けることができる。

⑤ 画面表示までの時間が、従来のＭＳＸに比べて速くなった。命令によっては、５倍程度速くなったものもあれば、10倍近く速く画面表示されるものもある。

OK…

⑥　ローマ字かな変換機能がある。

⑦　マウスやトラックボールなどの入力を読み取る機能が増えた。

⑧　32Ｋバイトのﾛ ＲＡＭディスクが増設できる。

⑨　時計内蔵型には、リアルタイム・クロック機能が付いた。

⑩　オプションでビデオ映像の取り込み機能を持っている。

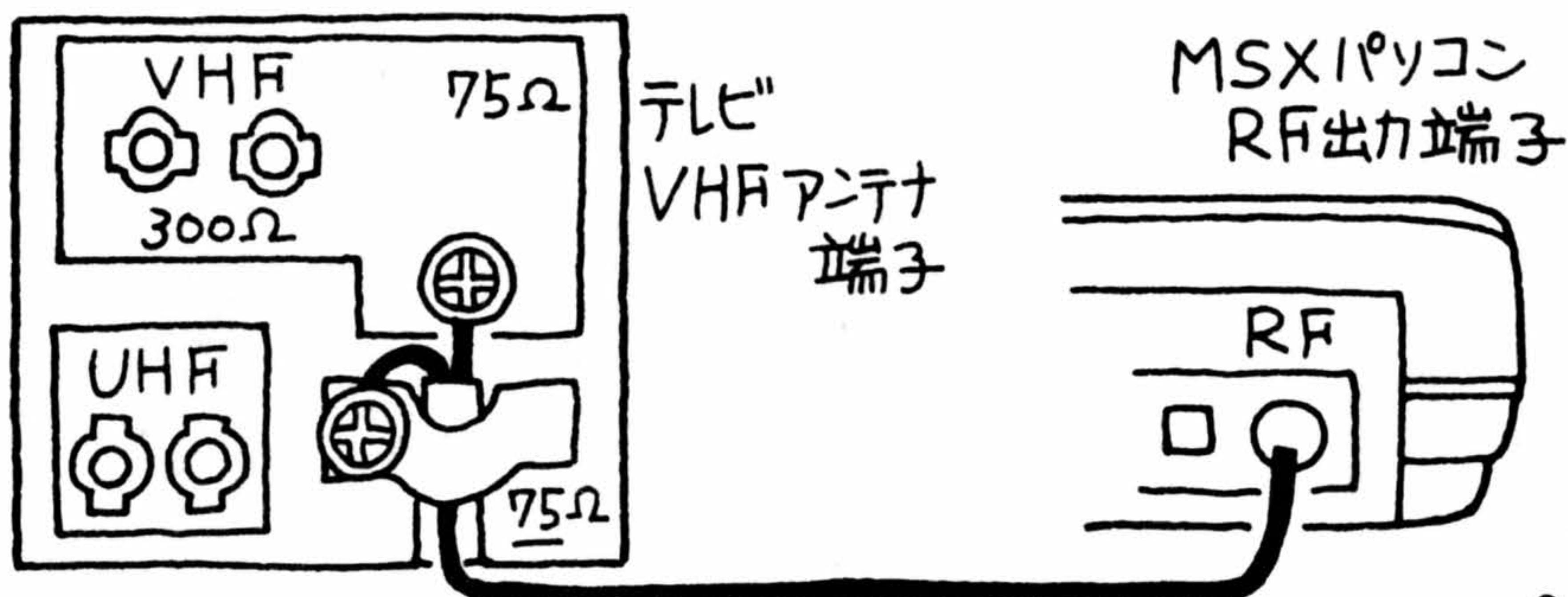
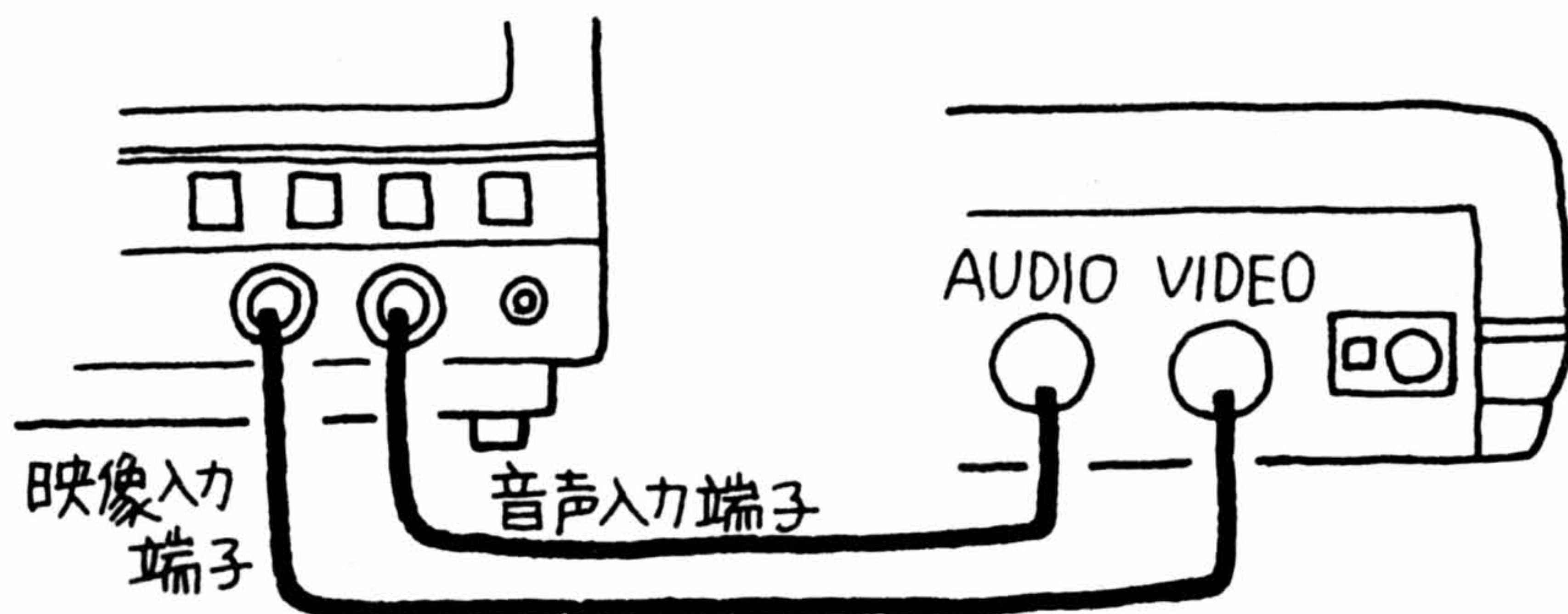
❺ＭＳＸタイプ・コンピュータ周辺機器とセッティング

機械を使用する前に、置いた場所の電源の接続や本体と他の周辺機器との接続をただしく行なう。

■ディスプレイ

パソコンにプログラムを実行させるとき、それを目に見えるようにしなければならない。そのために必要なものが出力装置であり、その代表的なものとしてＣＲＴディスプレイがある。ＣＲＴディスプレイは、ブラウン管(ＣＲＴ)をコンピュータに接続し、ブラウン管上にデータを表示し、また入力するものである。

ＭＳＸパソコンは、一般家庭用テレビに接続できるようになっており、本体のＣＲＴ出力内容に応じて次の三つの接続方法がある。

1．ＲＦ出力端子から一般家庭用テレビのアンテナ端子(75Ω)に内芯線を，入力端子へ外芯線を接続する。

2．ビデオ出力端子からテレビのビデオ入力端子に接続し、さらにサウンド出力端子からテレビの音声入力端子に接続する。

3．ＭＳＸタイプ・コンピュータのなかには、一部ＲＧＢ端子を持つものがあり、この機種にはＲＧＢ対応の専用テレビが必要である。接続方法は１と同様、出力端子と入力端子をつなぐ。

■プリンタ

もうひとつの出力装置。コンピュータに与えた仕事の命令やその実行結果はディスプレイで見ることができるが、それを記憶したり、記録、保存したりするために紙にプリント（印字）するための外部出力装置である。

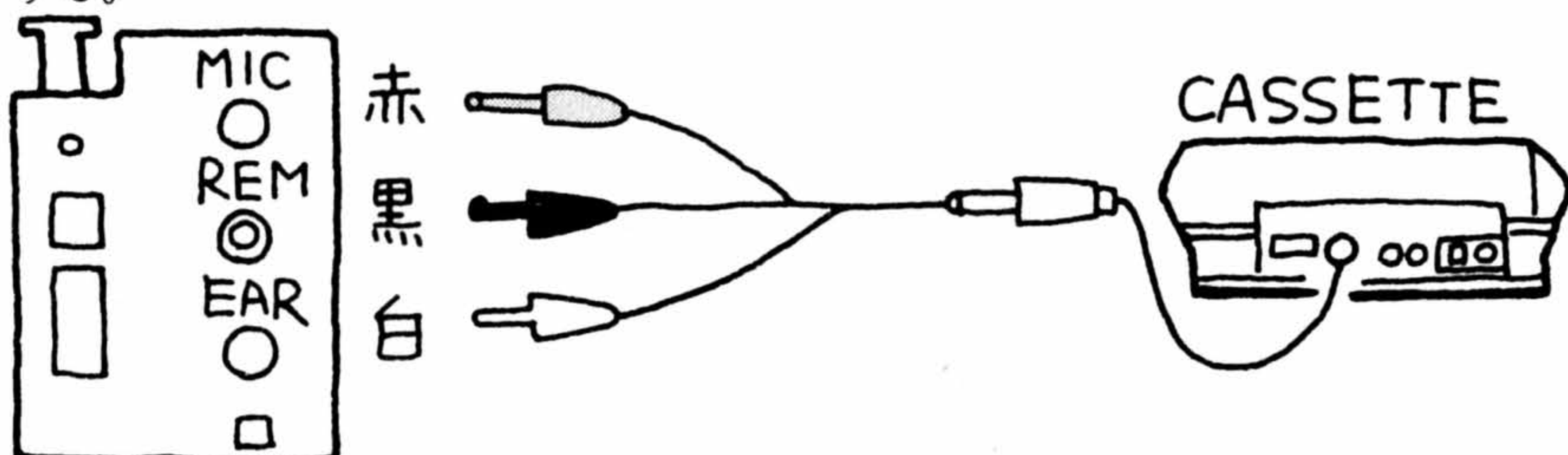
ＭＳＸパソコン専用のプリンタであれば問題ないが、それ以外の場合はコントロールコードの違いでグラフィックパターンなどが異なる場合がある。

プリンタには、ドットプリンタのほか、用途によってカラープリンタ、熱写方式などのプリンタがある。

プリンタ・インターフェイスは、機種によってオプションのものもある。

■カセット・レコーダ

カセット・レコーダは、パソコンの外部記憶装置として、コンピュータのメモリに保存されたプログラムやデータをテープに記録したり、その逆にあらかじめテープに記録されたプログラムやデータをメモリに読み込ませるときに使用する。

接続のしかたは本体のカセット入力端子にカセット・レコーダのＣＭＴ ＯＵＴ（EAR）白、ＣＭＴ　ＩＮ（MIC）赤、ＲＥＭＯＴＥ（リモート）黒、を接続する。

「赤いマイクイン」と覚えておくと接続に便利だ。また、リモート端子のないカセット・レコーダを使用する場合は、黒いプラグを空けておく。

■ジョイスティック

ゲームを行なうとき、キーボード上のカーソル・コントロールの上下左右キーを操作したり、また、たとえばミサイルを発射させるのにスペースキーを操作したりしてゲームを楽しむが、こうしたキーの代わりにジョイスティックを使えば、バーを上下左右にコントロールするだけで、ゲームをスムーズに動かすことができる。

ＭＳＸは統一規格であるため、どのジョイスティックを使用してもすべて作動する。ジョイスティックは、ＭＳＸ本体の側面に２個の接続端子を持っている。１台で使用する場合は、１の番号に接続する。機種によってはＡとＢになっているものもあり、そのときはＡに接続する。

■カートリッジ

ＭＳＸパソコンではさまざまなゲームカートリッジが発売されているが、カートリッジをスロットに差し込むときや抜き取るときは、本体電源を必ずＯＦＦにする。カートリッジを差し込み、電源をＯＮにする前に、カートリッジがしっかり本体に差し込まれているかどうかを確認する。また電子部品なので金属製の端子を水に濡らしたり、手であまり触れたりすることは避ける。

⑥フロッピーディスクとディスクドライブ

ＭＳＸ2のフロッピーディスクドライブには、コンピュータ本体に組み込まれている内蔵型と、本体とは別になっている分離型との２種類がある。

分離型の場合は、コンピュータ本体とは別にディスクドライブを購入して、接続する。また、分離型のフロッピーディスクには、１ドライブ用しかないので、２ドライブを必要とする場合は、さらにもう一台のディスクドライブを購入しなければならない。

一方、内蔵型はあらかじめコンピュータ本体に組み込まれているため、取り扱い方も便利で、また、１ドライブと２ドライブのふたつのタイプが用意されている。最近では、フロッピーディスク内蔵型が価格的にも安価になってきており、このタイプのＭＳＸ2が主流になっている。

■フロッピーディスクドライブの接続

内蔵型については問題ないが、分離型の場合はＭＳＸ2本体とディスクドライブを接続しなければならない。

接続には、必ずインターフェイスとケーブルを使用する。現在は、インターフェイスとケーブルが一体化されているものが多く、これを本体とディスクドライブに接続するだけでいいようになっている。

２ドライブにするためには、図２のようにＡドライブにＢドライブを接続する。このとき、インターフェイスには必ず両面、片面切り換えスイッチがついているので、１ドライブのときは片面に、２ドライブのときは両面に、スイッチを切り換える。これを行なわないと、うまく作動しないので注意する。

■フロッピーディスクおよびドライブ使用上の注意

ＭＳＸ2のディスクドライブでは、両面、片面の3.5インチマイクロフロッピーディスクが使用できる。ディスクの種類は、両面の場合は2ＤＤか2Ｄ、片面の

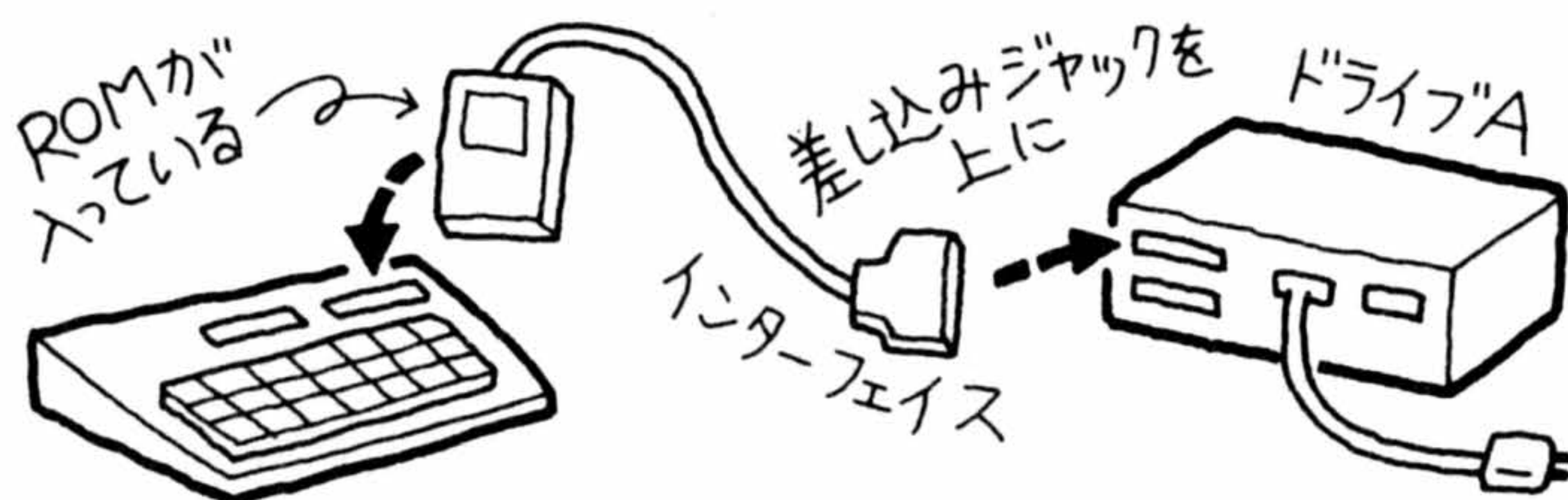



図1　1ドライブのときの接続

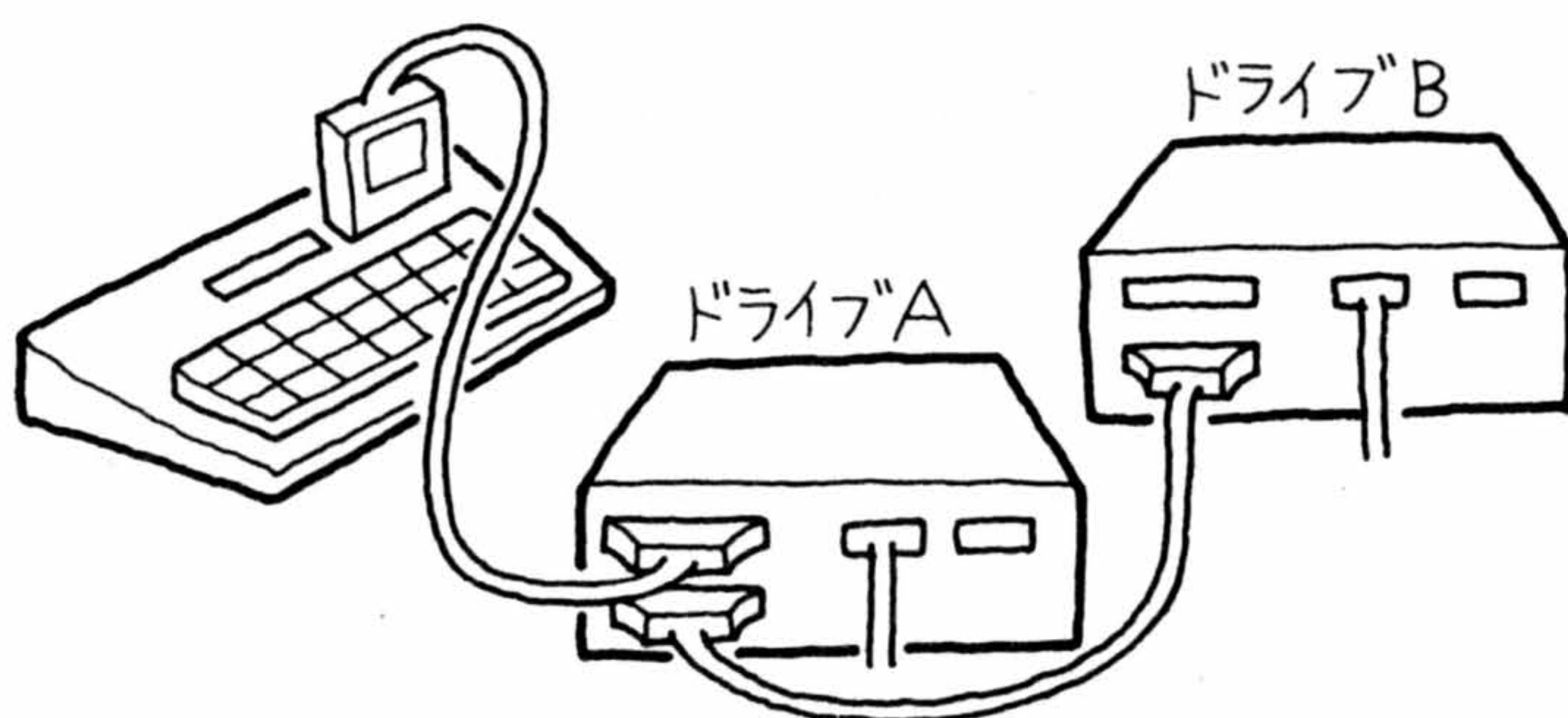



図2　2ドライブのときの接続

場合は1ＤＤが使用できるが、できれば2ＤＤを使用するといい。

ディスクをはじめて使う人は、まず取り扱い上の注意を十分覚えておくことが必要である。

■取り扱い、保管上の注意

フロッピーディスクには、その上部に金属が使用されているが、この金属部分には直接手を触れないようにする。フロッピーディスクがディスクドライブに挿入されると、この金属部分が左側にずれて動作する。そのため、たとえ目に見えない手の脂や汚れでも、金属部分についていると、ヘッドコンタクトに影響して、エラーの原因になりかねない。

直射日光やほこりも避けること。常にケースに入れて保管する習慣をつける。

また、テープと同じように磁気に非常に弱いので、ステレオのスピーカーやテレビの上などにはぜったいに置かない。

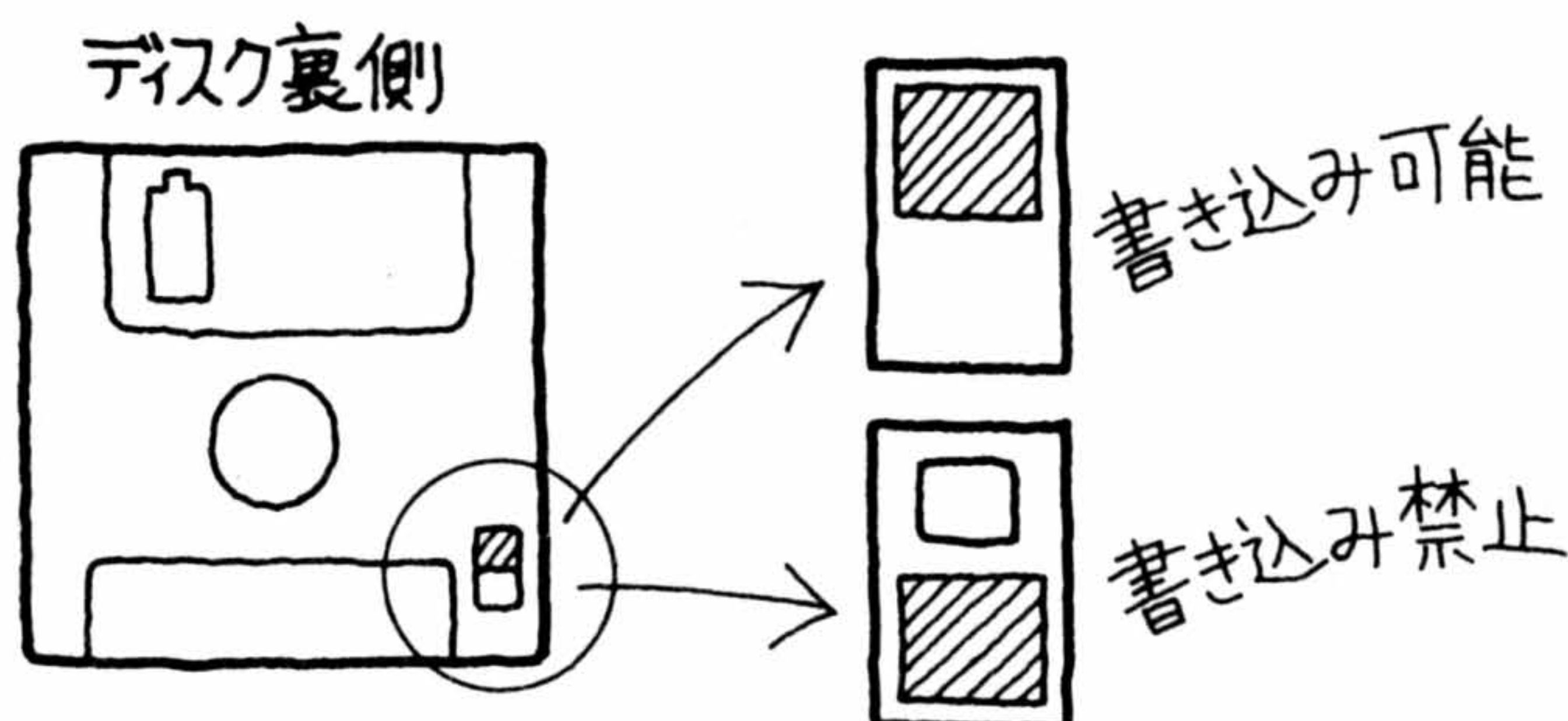
■ライトプロテクトタブ

ディスクの裏面の右下には、データを保護するライトプロテクトタブが付いている。このタブは、上下にスライドさせるようになっており、上にするとドライブへの書き込みができるようになり、下にすると書き込みが禁止される。

ライトプロテクトタブは、誤ってディスクのデータを消去したりしないよう保護するためにつけられているもので、たいせつなデータにはプロテクトをかけるか、このライトプロテクトタブを下にセットしておく。

ディスク裏側

書き込み可能

書き込み禁止

■ラベル

ディスクには、所定の位置にラベルを貼ることができ、ディスクの内容を記録しておくには便利である。ただし、何度も使って、ラベルを重ねて貼ると、ディスクに厚みができて、ディスクドライブを故障させる原因になる。必ず前のラベルをはがしてから、新しいラベルを貼るようにする。

■ディスクの入れ方

ディスクはまっすぐにドライブに入れる。入り口にぶつけないように注意しながら、カチャッと音がするまで挿入する。音がしない場合は、再度入れ直す。

■ドライブの作動ランプ

フロッピーディスクドライブには、２か所にランプがついている。ひとつはメインスイッチ作動ランプ（パワーランプ）で、電源ＯＮの状態でランプが点灯する。

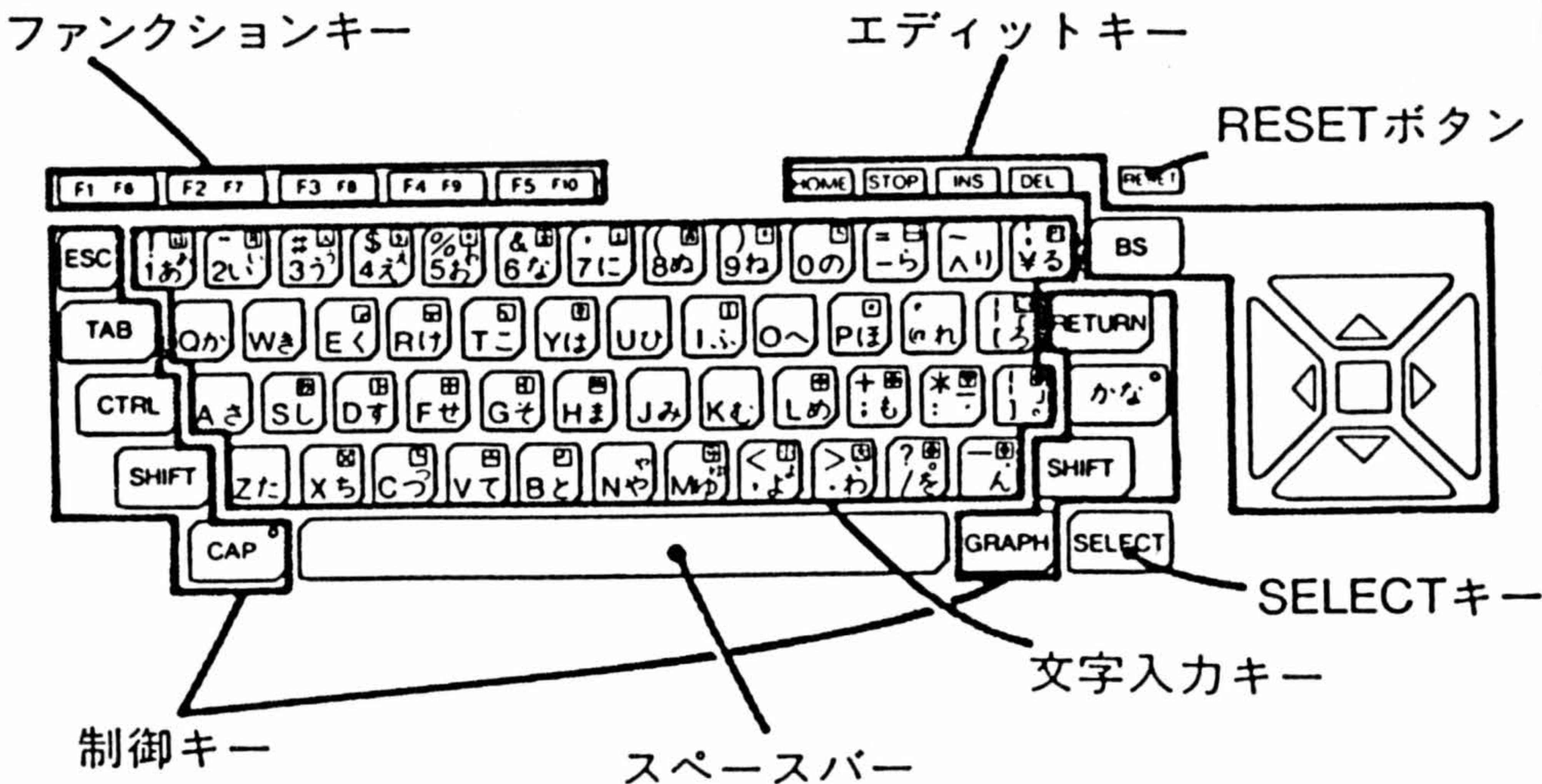
もうひとつは、ディスクの内容を読んだり、ディスクにデータを書き込んでいるときに点灯する作動ランプである。

特に注意したいのは、この作動ランプが点灯しているときである。このときにはぜったいにディスクの出し入れはできない。むりに出したり入れたりすると、ディスクが破損するだけでなく、ディスクドライブのヘッド部分を壊してしまう。

❼キーボードを知る

キーには、フルキー、タッチキーがある。タッチキーはよく電卓でみられる押しボタン式のキーである。

カナキーの配列も、ＪＩＳ配列のものと、50音配列の２つのタイプにわけられる。

ファンクションキー
エディットキー
RESETボタン
SELECTキー
文字入力キー
制御キー
スペースバー

●特殊記号の呼び方

記号	呼び方	記号	呼び方
!	エクスクラメーション	゛	濁点
”	ダブルクォーテーション	「	左カギカッコ
#	ナンバーサイン（イゲタ）	[	左カギ
$	ダラーサイン（ドルマーク）	゜	半濁点
%	パーセント	+	プラス
&	アンパーサンド	;	セミコロン
’	アポストロフィー	*	アスタリスク
(	左カッコ	:	コロン
)	右カッコ	]	右カギ
=	イコール（等号）	」	右カギカッコ
~	ウェーブバー（波横棒）	<	不等号（小）
-	マイナス	,	コンマ
^	アップアロー（ヤマガタ）	>	不等号（大）
¥	エンサイン	.	ピリオド
ー	バー（長音）（ハイフォン）	?	クェスションマーク
@	アットマーク（サイン）	/	スラッシュ

●キーの切り換え機能

キーボードを見ればわかるように、１つのキーが何種類もの役割を持っている。このいろいろなキーの役割を切り換えるためのキーが、以下４つのキーである。

SHIFT（シフトキー）………小文字、大文字の切り換えや特殊記号を入力するためのキー

大文字キー＋SHIFT──→小文字入力

小文字キー＋SHIFT──→大文字入力

CAPS LOCK（キャプスロックキー）…………このキーを単独で使った場合は、英文字キーの大文字、小文字の切り換えを行なう。キーを押した状態で英大文字になる。

GRAPH……グラフィック記号を入力するためのキー。
(グラフィックキー)

カナ……ひらがなを入力するキー。他のキーとの組み合わせで、ひらがな小文字、カタカナ、カタカナ小文字が入力される。

カナキー ——→ひらがな入力
カナ+SHIFT ——→ひらがな小文字入力
カナ+CAPS LOCK ——→カタカナ入力
カナ+CAPS LOCK+SHIFT——→カタカナ小文字入力

●特殊キー

F1〜F5
F6〜F10
(ファンクションキー)
……ファンクションキーと呼ばれ、主に、たくさん使う言語をワンタッチで入力できるようにしている。
ほとんどの機種はキーボードの左上に5個の長方形が並び、キー上にはF1・F2・F3・F4・F5と書かれている。そのままキーを押すと、各キーがそれぞれ持っている1〜5の機能が働き、SHIFTキーを押しながらキーを押すと、それぞれもうひとつずつ持っている6〜10までの機能が働く。

STOP……プログラム実行中やゲーム中にこのキーを押すと、実行中のプログラムはストップして一時停止の状態になる。もう一度STOPキーを押せば、またすぐに実行が再開する。この一時停止状態のときは入力はできない。キー入力する場合は、CTRLキーを押しながらSTOPキーを押す。このとき処理を再開するのには、CONT, RETURNとキー入力する。
(ストップキー)

INS……このキーを押すと、カーソルの形が“■”から“▬”へ変わり、挿入モードになる。挿入モードで文字キーを押すと、
(インサートキー)

カーソルの直前に文字が挿入され、押されたうしろの文字や数字は消えずに右へ移動する。再びこのキーを押すかカーソル移動キーまたはRETURNキーを押すと、カーソルはもとの形に戻り、挿入モードが解除される。

DEL……………このキーを押すとカーソルの位置にある文字が削除され、
(デリートキー)　その右側の文字列すべてが左に一桁移動する。

HOME/CLS…………このキーを押すと、カーソルが画面左上端に戻る。また、
(ホームクリアキー)　SHIFTキーを押しながらHOME/CLSキーを押すと、画面をきれいに掃除しながら、カーソルが画面左上端に移動する。

ESC……………このキーを押すと、コントロールコード27が入力できる。
(エスケープキー)

SELECT……このキーを押すと、コントロールコード24が入力できる。
(セレクトキー)

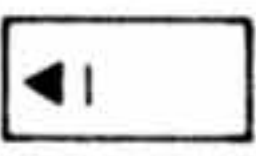
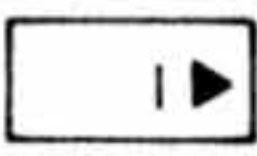
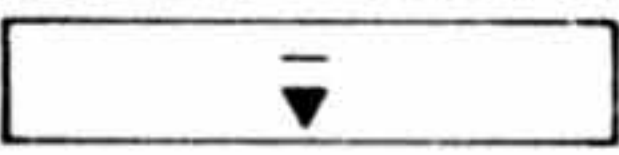
BS………………このキーを押すと、カーソルの直前の文字が削除され、そ
(バックスペースキー)　の右側の文字列すべてが左に一桁移動する。

TAB……………このキーを押すと、カーソルがTAB位置 1, 9, 17, 25
(タブキー)　と8桁ごとにステップしその間の文字列は空白になる。主に表を作るときなどに使用する。

RETURN……このキーを押すと、キー入力した内容をパソコンに伝える。
(リターンキー)

CTRL…………このキーを押しながら他のキーを押すと、次のように特殊
(コントロールキー)　な働きをする。

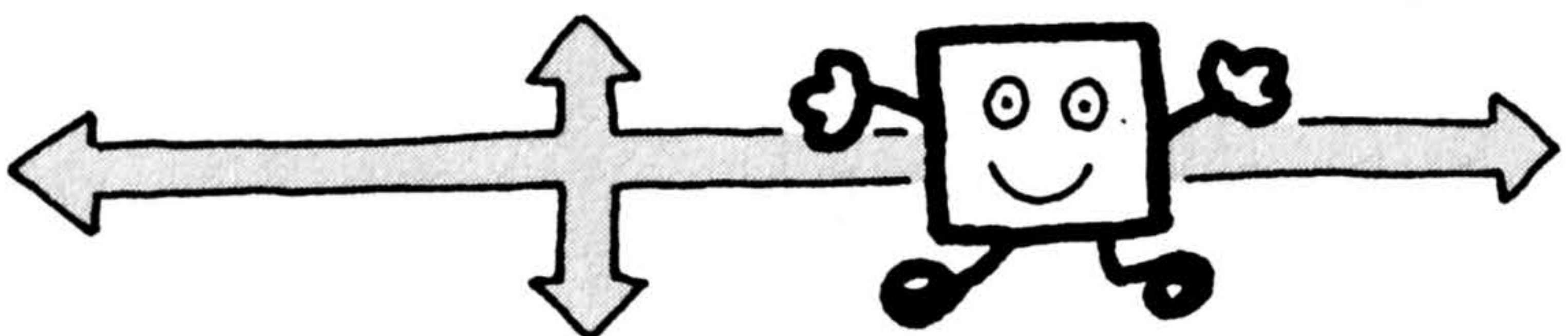
●CTRLキーと他のキーの組み合わせ

CTRL+A……グラフィックキャラクタの入出力時のヘッド。CHR$(1)

CTRL+B……ひとつ前の単語の先頭へカーソルを移動する。

CTRL+C……入力待ち状態を終了し、次の行の最初にカーソルを移動する。

CTRL+E……カーソル以下を削除する。
CTRL+F……ひとつ後の単語の先頭へカーソルを移動する。
CTRL+G……スピーカーをピッと鳴らす。（BEEP文と同じ）
CTRL+H……BSキーを押したのと同じ働きをする。
CTRL+I……TABキーを押したのと同じ働きをする。
CTRL+J……行送りをし、カーソルを１行下へ移動する。
CTRL+K……HOME/CLSキーを押したのと同じ働きをする。
CTRL+L……SHIFTキーを押しながらHOME/CLSキーを押したのと同じ働きをする。
CTRL+M……RETURNキーを押したのと、同じ働きをする。
CTRL+N……カーソルを行の一番最後に移動する。
CTRL+R……INSキーを押したのと同じ働きをする。
CTRL+U……カーソルの置いてある一行を画面から削除する。
CTRL+X……SELECTキーを押したのと同じ働きをする。
CTRL+[……ESCキーを押したのと同じ働きをする。
CTRL+¥……カーソルを右へ移動する。
CTRL+]……カーソルを左へ移動する。
CTRL+^……カーソルを上へ移動する。
CTRL+－……カーソルを下へ移動する。
CTRL+STOP……コンピュータの作動を強制的に停止させて、キー入力の受け付けを待つ状態にする。

●カーソル移動キー

▲ ◀ ▶ ▼……カーソルを上下左右それぞれに移動するのに使用する。

⑧MSX-DISK BASIC スタート

ディスクドライブ内蔵型コンピュータは、通常、電源を入れればＭＳＸ-ＤＩＳＫ　ＢＡＳＩＣが作動するようになっているが、コンピュータ本体にあとからディスクドライブを接続して使う分離型の場合は、作動する形態が多少異なる。

ディスクドライブ分離型の場合は、必ず別売のインターフェイスケーブルで本体とディスクドライブを接続する。このインターフェイス部分のカートリッジに、ＭＳＸ-ＤＩＳＫ　ＢＡＳＩＣのＲＯＭ（Read Only Memory＝読み出し専用メモリ）が内蔵されており、本体と接続することによって、ＤＩＳＫ　ＢＡＳＩＣが作動する仕組みになっているのである。

さて、ＤＩＳＫ　ＢＡＳＩＣを立ち上げると、バージョン2.0では、画面に次のように表示される。

```
MSX BASIC Version 2.0
Copyright 1985 by Microsoft
xxxxx Bytes free
DISK BASIC Version 1.0
```

このＤＩＳＫ　ＢＡＳＩＣは、ディスクドライブを使用できるようにするためのＢＡＳＩＣで、ディスク内の管理をするディスク・オペレーティング・システム（ＤＯＳ）とは異なるので、注意する。

■バックアップの方法

ＭＳＸ－ＤＯＳシステムディスクのように、１枚のディスクしかついていないものは、ディスクを使う前に、必ずコピーし、コピーしたほうのディスクを使うようにしたほうがよい（これを、「バックアップをとる」という）。万一、ディスク内のシステムが壊れたり、誤って消してしまっても、オリジナルをとっ

てあれば、またそれをコピーして使えばいいので、安心である。
　ここでは、ＭＳＸ－ＤＯＳシステムディスクのコピーを例にとって、ディスクのバックアップ方法を説明する。

①ディスクのフォーマット

　まず新しいディスクをフォーマットする。未使用の新しいディスクは、フォーマットしないと使える状態にならないので必ず行なうこと。ＦＯＲＭＡＴ命令は、ＭＳＸ－ＤＯＳコマンドのＦＯＲＭＡＴの項を参照。

②ＭＳＸ－ＤＯＳのＳＹＳ（システム）のコピー

　次に、ＭＳＸ－ＤＯＳのＳＹＳ（システム）をコピーする。このとき、ディスクドライブが２台ついているタイプ（２ドライブ）と１台しかないタイプ（１ドライブ）ではコピーする方法が異なる。

・２ドライブ以上使える場合のコピー

　２ドライブ以上使える場合は、システムディスク（ここではＭＳＸ－ＤＯＳのシステムディスク）をドライブＡに入れて、フォーマット済みのディスクをドライブＢに入れる。
　ディスクがしっかりとセットされていることを確認したら、

　　　　ＣＯＰＹ　Ａ：＊．＊　Ｂ：　ＲＥＴＵＲＮ

と入力する。すると、すべてのシステムファイルがコピーされる。
　また、１システムファイルずつコピーしたい場合は、次のように、

　　　　ＣＯＰＹ　Ａ：ＭＳＸＤＯＳ　．ＳＹＳ　Ｂ：

と入力して ＲＥＴＵＲＮ キーを押すと、コピーが行なわれる。
　この方法で、ＣＯＭＭＡＮＤ．ＣＯＭをはじめ、各システムファイルをコピーする。

　　　　ＣＯＰＹ　Ａ：ＣＯＭＭＡＮＤ　．ＣＯＭ　Ｂ：Ｂ　ＲＥＴＵＲＮ

・１ドライブでコピーする場合

　１ドライブしかない場合のコピー方法は、少し時間がかかる。また、１台でＡドライブとＢドライブの両方を兼ねなければならず、ディスクの出し入れが多くなる。このとき、ディスクを破損させないように注意する。
　１ドライブでコピーを行なうときは、まずシステムディスクをドライブに挿

入し、2ドライブのときと同じように、

COPY A：MSXDOS .SYS B：

と入力し、RETURN キーを押す。するとシステムディスクの内容がメモリに読み込まれ始め、読み込みが終了すると次のメッセージが画面に表示される。

Insert diskette for drive B：
and strike a key when ready

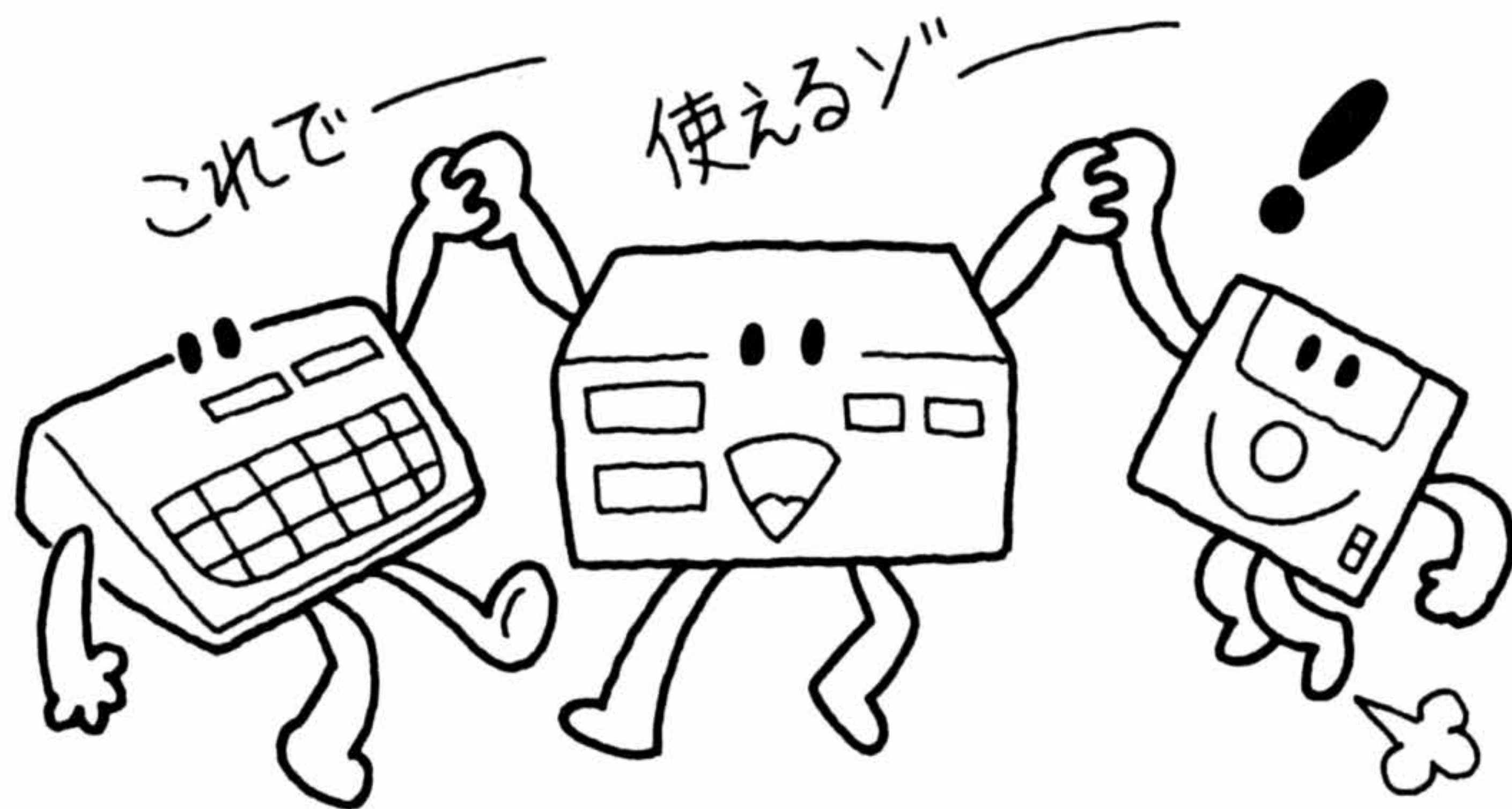
この表示が出たら、ドライブからシステムディスクを取り出し、フォーマット済みの新しいディスクを挿入する。正しく挿入されているかを確認して、キーボード上のどのキーでも押すと、今度はメモリからディスクの書き込みが始まる。

コピーが終了すると、

1 File copied

と表示されて、1つのシステムファイルがコピーされたことを確認できる。

1ドライブの場合は、このようにして、何回もコピーを繰り返していく。

1ドライブタイプでコピーをとるときは、システムディスクのライトプロテクトタブは、書き込み禁止のために必ず下にさげておく。

⑨MSX・BASICの演算機能

ＢＡＳＩＣの演算機能には①算術演算②関係演算③論理演算④関数演算⑤文字列演算の５種類がある。

●算術演算

算術演算は数値の四則計算である。記述計算での加減乗除に使用する＋－×÷の記号の代わりに＋－＊／の関係演算子を使用する。算術式の中には、文字定数や文字変数を含むことはできない。

（算術演算子）

^（べき乗）	指数演算で、たとえば 5^{10} なら5^10である。
－（負号）	数が負であることを示す。
＊（アスタリスク）	乗算に使用する。たとえば５×６なら５＊６である。
／（スラッシュ）	実数の除算に使用する。たとえば10÷2なら10／2である。
￥（エンサイン）	整数除算で／の代わりに使用する。このとき結果は整数部分だけが取り出される。たとえば、10÷3は10￥3とし、結果は3である。
ＭＯＤ	除算の剰余をだすときに使用する。結果は整数の除算のあまりである。たとえば、13.3　ＭＯＤ　4の結果は１である。
＋（プラス）	加算に使用する。
－（マイナス）	減算に使用する。

さまざまな演算子が１本の式に混在したとき、計算の優先順位は上表の通りだが、＊／、＋－の場合は優先順位は同等だから、先に現れた順に計算する。

また、優先順位を変更する場合は、はじめに演算させる順に（　）でくくる。

●関係演算

２個の数値または文字列を比較する。関係演算の結果は真（-1），偽（０）で求められる。

（関係演算子）

＝	双方が等しい。たとえばｘ＝ｙと使用する。
＜＞，＞＜	等しくない。たとえばｘ＜＞ｙ，ｘ＞＜ｙと使用する。
＜	小さい。たとえばｘ＜ｙと使用する。
＞	大きい。たとえばｘ＞ｙと使用する。
＜＝，＝＜	小さいか等しい。たとえば、ｘ＜＝ｙ，ｘ＝＜ｙと使用する。
＞＝，＝＞	大きいか等しい。たとえば、ｘ＞＝ｙ，ｘ＝＞ｙと使用する。

●論理演算

論理演算は、複数の条件を調べたり、ビット操作やブール計算に用いられる。論理演算は特に機械語の操作などで使用される。ここでは演算子と意味だけをあげておく。

ＮＯＴ	否定（not)。ＮＯＴ　Ｘと使用する。
ＡＮＤ	論理積（and)。Ｘ　ＡＮＤ　Ｙと使用する。
ＯＲ	論理和（or)。Ｘ　ＯＲ　Ｙと使用する。
ＸＯＲ	排他的論理和（exclusive or)。Ｘ　ＸＯＲ　Ｙと使用する。
ＩＭＰ	包含（implication)。Ｘ　ＩＭＰ　Ｙと使用する。
ＥＱＶ	同値（equivalence)。Ｘ　ＥＱＶ　Ｙと使用する。

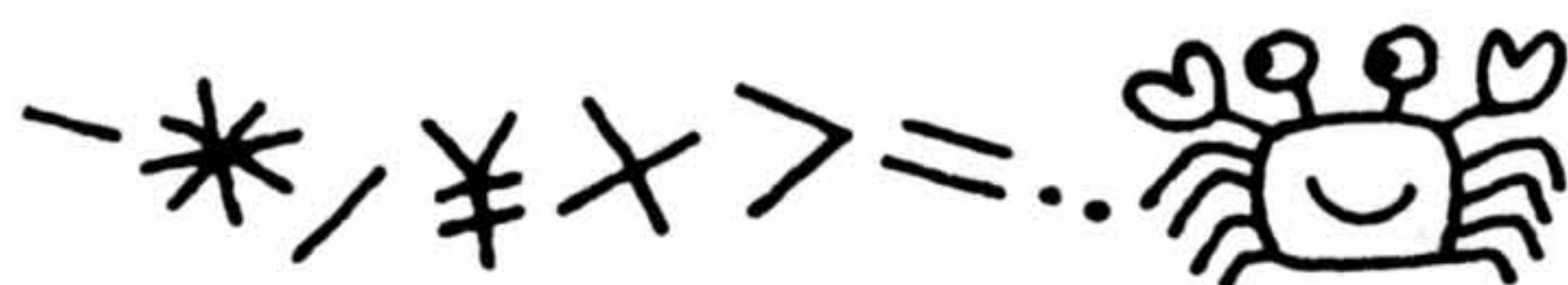
●関数

与えられた引数に対してある決まった演算の値をあらかじめ持っている。

●演算の優先順位

演算機能の各種演算子が１本の数式の中に混在したとき、各演算子の優先順位は次のようになる。

①（　）の中　②関数　③べき乗（指数）　④負号（－）　⑤＊，／⑥￥　⑦ＭＯＤ　⑧＋，－　⑨関係演算子（＜，＞，＝など）　⑩ＮＯＴ　⑪ＡＮＤ　⑫ＯＲ　⑬ＸＯＲ　⑭ＩＭＰ　⑮ＥＱＶ

－＊／￥＋＞＝..

MSX-DOS
入　門

❶ MSX-DOSスタート

ＭＳＸ－ＤＯＳ（ディスク・オペレーティング・システム）は、ディスク内のファイルの管理をするためのもので、プログラムを作成したり変更したりすることはできない。

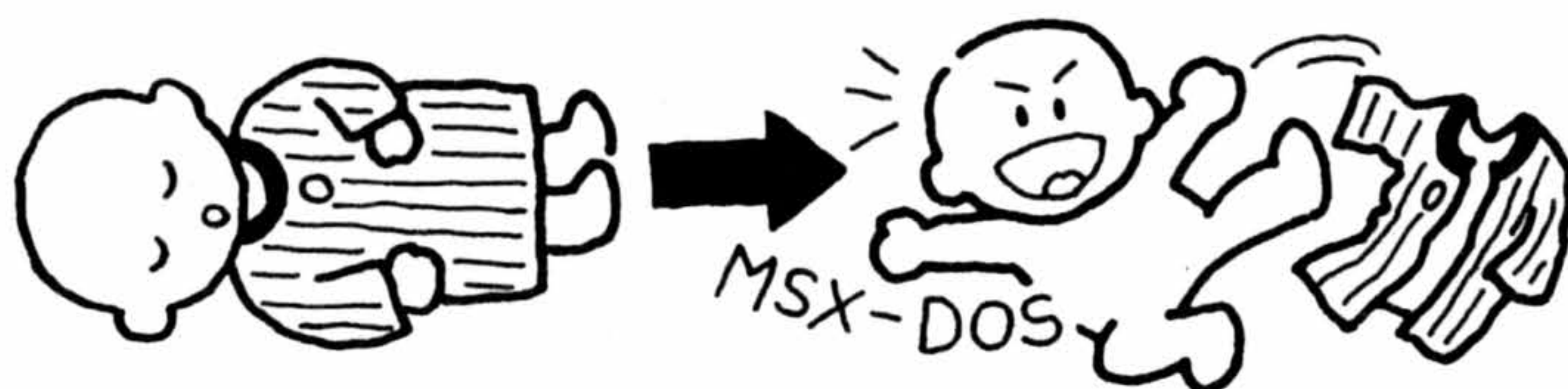
たとえば、ディスクをコピーしたり、新しいなまのディスクを使用できる状態にするためのフォーマットは、このＭＳＸ－ＤＯＳの管理のもとにある。

このＤＯＳは、起動時、64Ｋバイト以上のＲＡＭ（Random Access Memory＝書き込み、読み出しが自由にできるメモリ）が必要で、それ以下のＲＡＭ容量では作動しないので注意する。64Ｋバイト以上のＲＡＭ容量がないときは、別にＲＡＭボードを購入して、増設する。

通常、ディスクドライブが２つある場合には、ＭＳＸ－ＤＯＳのディスクは必ずドライブＡに挿入して起動させる。外部ドライブを使う場合には、ディスクをドライブＡに入れて、電源を入れると作動する。

ＭＳＸ－ＤＯＳが起動されると、「Ａ>」というコマンド命令のサインが表示されて、入力待ちの状態になる。これは、Ａドライブが優先ドライブであることを意味している。

この「Ａ>」は、Ａのディスクドライブを使用しているという意味であり、「>」の表示をプロンプトという。「Ｂ>」と表示されれば、２つあるディスクドライブのうちのＢドライブを使用しているということである。

Ａ>のプロンプトが表示されれば、ＭＳＸ－ＤＯＳが起動されたことになるので、各コマンドによってディスク管理を行なう。

❷ディスクエラー

ディスク作動中に、ディスクに関するエラーが起こると、次のメッセージが画面上に表示される。

xxx error xxxxx drive x
Abort, Retry, Ignore?

このときのコマンドには、A、R、Iという3つのコマンドが用意されており、その働きは次のとおりである。

A………プログラムの実行をその場で中止して、プロンプト表示待ちになる。

R………再度同じ処理を行なう。

I………そのエラーの発生した部分を無視して、そのまま先に進む。通常は、このコマンドは使用しないほうがよい。

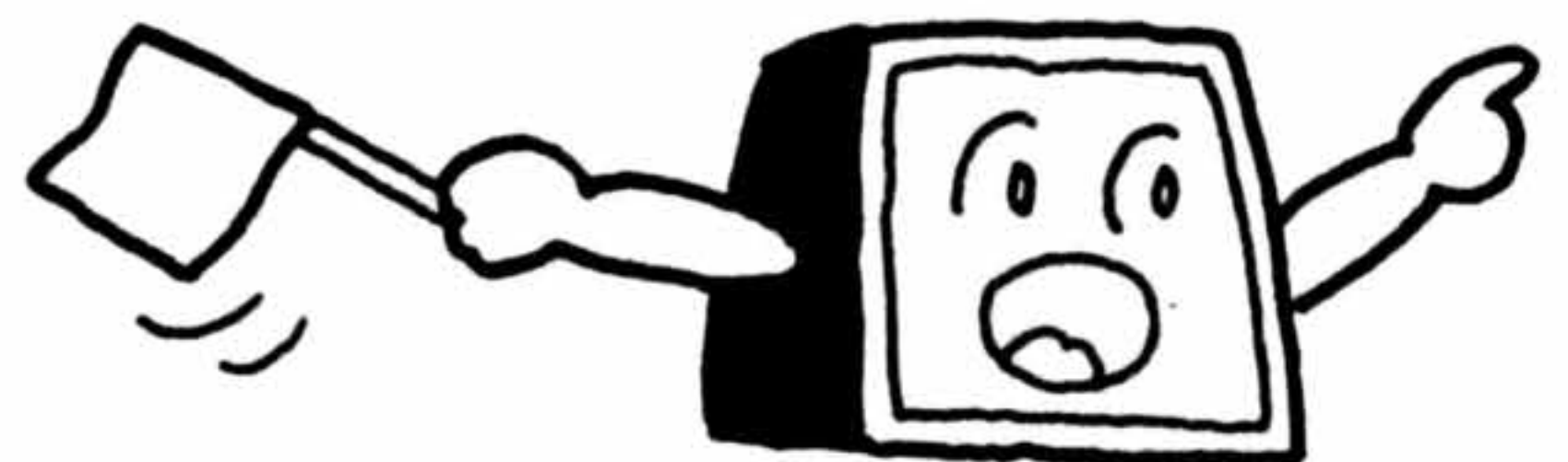
❸ワイルドカード

コピー命令で＊という記号を使用したが、これはワイルドカードといって、ファイル名やドライブのタイプを指定するときやコマンドを入力するときに、？（クエッションマーク）や＊（アスタリスク）をその省略形として使用できるものである。

「？」は、すべての1文字の代わりになる。

COPY A:MSX???.SYS B:

「＊」は、すべての文字列に有効で、その文字列は何文字でも可能である。

COPY A:M*.SYS B:

④MSX-DOS命令

BASIC（ベーシック）

MSX-DISK BASICを指定する

ＭＳＸ−ＤＯＳからＭＳＸ−ＤＩＳＫ　ＢＡＳＩＣを指定する命令である。

ＢＡＳＩＣのプログラムを指定してＢＡＳＩＣを起動させ、その後プログラムをロードして実行させる。

書式 ＢＡＳＩＣ ［(ドライブ名)］：［ファイル名．タイプ名)］

説明 ドライブ名を指定しない場合は、現在使用中のディスクドライブとなる。

ファイル名は、８文字以内の文字列に限り有効である。また、タイプ名は、３文字以内の文字列で、ＢＡＳタイプにつき有効である。

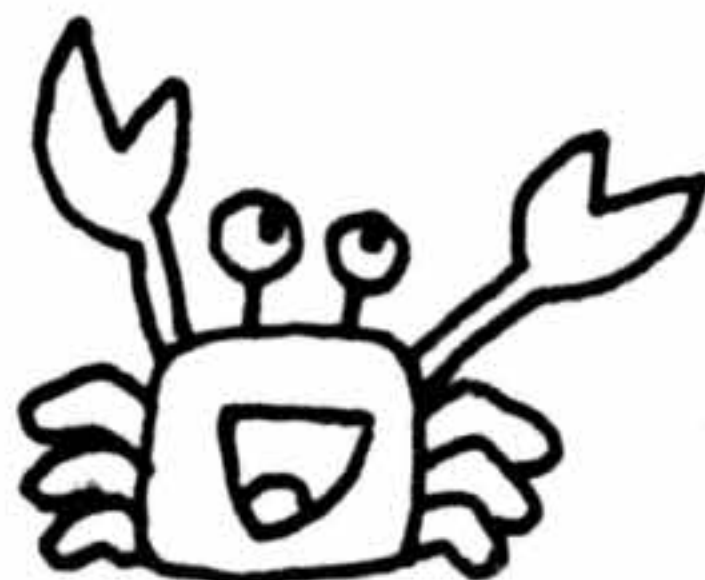
書式例 ＢＡＳＩＣ　Ａ：ＴＥＳＴ1 .ＢＡＳ

これを実行すると、コンピュータ内部のスロットを自動的に切り換えて、ＲＯＭ内のＤＩＳＫ　ＢＡＳＩＣが起動される。

ＤＩＳＫ　ＢＡＳＩＣが起動したら、次にドライブＡの中にあるディスクファイル名（ここではＴＥＳＴ1．ＢＡＳ）を見つけて、プログラムをロードし実行する。

また、ＭＳＸ−ＤＯＳとＭＳＸ−ＤＩＳＫ　ＢＡＳＩＣではメモリマップが異なるが、ＭＳＸ−ＤＯＳからＭＳＸ−ＤＩＳＫ　ＢＡＳＩＣに移行したときのメモリマップに自動的に切り換えられる。

この逆で、ＭＳＸ−ＤＩＳＫ　ＢＡＳＩＣから、ＭＳＸ−ＤＯＳに戻ることもできる。命令は、ＣＡＬＬ　ＳＹＳＴＥＭ文で、ＢＡＳＩＣ中で使用する。

ＣＡＬＬ　ＳＹＳＴＥＭ文は、ＢＡＳＩＣを起動させてからＭＳＸ−ＤＩＳＫ　ＢＡＳＩＣに移動したときだけに使用できる命令なので、特に注意する。

FORMAT（フォーマット）

■新しいディスクを使用できる状態にする

新しいディスクは、フォーマットを行なわなければ使用できない。（イニシャライズするともいう）。使用する前に、必ずＭＳＸ－ＤＯＳを起動してフォーマットを実行する。

ＭＳＸ－ＤＯＳとＭＳＸ－ＤＩＳＫ　ＢＡＳＩＣのフォーマットは同じなので、ＭＳＸ－ＤＯＳでフォーマットしたものはＢＡＳＩＣでも使用できるようになっている。

書式

```
（1）FORMAT
（2）CALL  FORMAT（省略形は_FORMAT）
```

説明　（1）はＭＳＸ－ＤＯＳからの命令で、（2）はＭＳＸ－ＤＩＳＫ　ＢＡＳＩＣからの命令である。書式（1）（2）とも同じＦＯＲＭＡＴ文を作動させ、フォーマットを行なうので、どちらで行なってもかまわない。

ＦＯＲＭＡＴを実行すると、画面上には、

```
Drive name ?（A,B）
```

と表示される。これは、Ａ、Ｂどちらのドライブでフォーマットを行なうのかを選択するものである。ここで、ＡかＢをキーボードから入力する。ただし、ドライブが１台しかない場合は、常にＡを選択する。

ＭＳＸ2のディスクドライブには、両面と片面の両方のディスクを使える機種もある。このときには、自動的に画面上に、

```
1－Single sided, 9 sectors
2－Double sided, 9 sectors
```

と表示される。片面のディスクをフォーマットする場合には１を、両面のディスクを使用しているときは２を、それぞれ選択する。

指定が終わると、

```
Strike a key when ready
```

と表示されるので、ディスクドライブに新しいディスクを入れて、キーを何か1つ押す。これで自動的にフォーマットが開始される。

フォーマットが終了すると、画面に、

Format complete

と表示されて、ドライブAを使用しているときにはA＞とプロンプトが表示されて、キー入力待ちとなり、フォーマットを終了する。

DIR（ディレクトリー）

■ディレクトリー内のファイルを表示する

書式 DIR ［ドライブ名］：［ファイル名．タイプ名］［／P］［／W］

説明 ドライブを指定することによって、そのディレクトリー内のファイル名とそのファイルの大きさを表わすバイト数、そして最後に修正を受けた日付けが表示される。

さらに、時計機能を持っているコンピュータで、1行に36文字以上の表示が可能な機種の場合には、日付けのあとに時刻が表示される。

書式例

（1）DIR…………現在使用中のドライブ上のディスクにあるファイルを、すべて画面に表示する。

（2）DIR A：…ドライブAのディスクにあるすべてのファイルを、画面に表示する。

（3）DIR A：TEST1．DAT
………ドライブAのディスク内にあるTEST1．DATのファイルを探し出して、画面に表示する。

（4）DIR A：TEST1．*
………ドライブAのディスク内にあるTEST1のファイル名を、すべて画面に表示する。

DIR A：*．DAT
………ドライブAのディスク内にあるタイプ名DAT

のファイルを、すべて画面に表示する。

このように、ファイル名やタイプ名の代わりに文字列の代わりとなる「?」や「*」のワイルドカードを使うことができる。

(5) DIR /P…現在使用中のドライブ上のディスクにあるすべてのファイルを画面に表示する。画面表示範囲を超えるといったん表示が止まる。続いてファイルを見るときは、キーをどれか1つ押すと、残りのファイルが表示される。

(6) DIR /N…現在使用中のドライブ上のディスクにあるすべてのファイルを画面に表示する。この場合は、ファイル名だけを、1行に詰められるだけ詰めて表示するので、ファイルが多いときには便利である。

DELETE(デリート)
ERASE(イレーズ)

■指定されたディスク上のファイルを消去する

書式

```
DEL  [ドライブ名]:[ファイル名.タイプ名]
ERASE  [ドライブ名]:[ファイル名.タイプ名]
```

説明 DEL、ERASEともに、指定されたディスク上のファイルを消去する。ファイル名とタイプ名には、「?」(クエッションマーク)と「*」(アスタリスク)のワイルドカードが使える。

書式例 (1) DEL A:TEST1.DAT
ERASE A:TEST1.DAT

ドライブAのTEST1.DATのファイルを探し出して存在すればそのファイルを消去する。

(2) DEL A:TEST1.*

ERASE　A：TEST1.＊

ドライブAのTEST1のファイル名がなんであっても、そのすべてにあてはまるファイルを消去する。

（3）DEL　A：＊.＊

ドライブAのファイルすべてを消去する。画面上に"Are you sure ?（Y/N)"と表示し、消去してよいかどうかの確認を求めてくる。「Y」を入力すると、ドライブAのすべてのファイルを消去する。

COPY（コピー）

■ディスク上のファイルをコピーする

書式

COPY　[ドライブ名1]：[ファイル名1．タイプ名1]　[ドライブ名2]：[ファイル名2．タイプ名2]

説明　ディスクからディスクへのファイルコピーを行なう。ディスクは同一ディスクでも、他のディスクでもコピーを行なう。

ただし、同一ディスクの場合は、同一ドライブ上では同じファイル名ではコピーできないので、ファイル名を変更しなければならない。

ファイル名やタイプ名には、「？」（クエッションマーク）や「＊」（アスタリスク）のワイルドカードが使用できる。

書式例　（1）　COPY　A：TEA．DAT　B：

ドライブAの中にあるファイル名TEA．DATを、ドライブBに同一ファイル名でコピーする。

（2）　COPY　A：TEA．C＋XYZ．C　B：TEX．C

ドライブAの中のファイル名TEA．CとXYZ．Cを合併して、ドライブBのディスクにTEX．Cというファイル名でコピーする。

（3）　COPY　AB．DAT　CD．PRS

使用ドライブにあるファイル名AB．DATを、同一ファ

イルにCD．PRSというファイル名でコピーする。

（4）　COPY　A：*．DAT　B：

ドライブAにあるDATというタイプ名のすべてのファイルを、ドライブBに同じファイル名でコピーする。

DATE（デート）

日付けを表示し、変更する

書式　DATE　[年－月－日]

説明　年月日を変更したいときに使用する。それぞれの間の「－」（ハイフン）は、「／」（スラッシュ）でも代用できる。

DATEを入力してRETURNキーを押すと、次のメッセージが画面に表示される。

```
Current date is 曜日 年－月－日
Enter new date：
```

年月日を入力するとき、1988年としたい場合は、19を省略して88と入力する。

（例）88／5／30

MSX－DOSでは、月の大小（30日と31日の月）やうるう年も判別できるようになっている。

入力条件が合わない場合には、

```
Invalid date
Enter new date：
```

のメッセージが表示されるので、再度キー入力する。

REN（リネーム）

■指定したドライブの中のファイル名を変更する

書式

REN ［ドライブ名］:［ファイル名．タイプ名］〔新ファイル名．タイプ名〕

説明 指定されたディスクドライブ中のファイル名やタイプ名を変更するときに使用する。

ファイル名やタイプ名には、「?」（クエッションマーク）や「*」（アスタリスク）のワイルドカードを使うことができる。

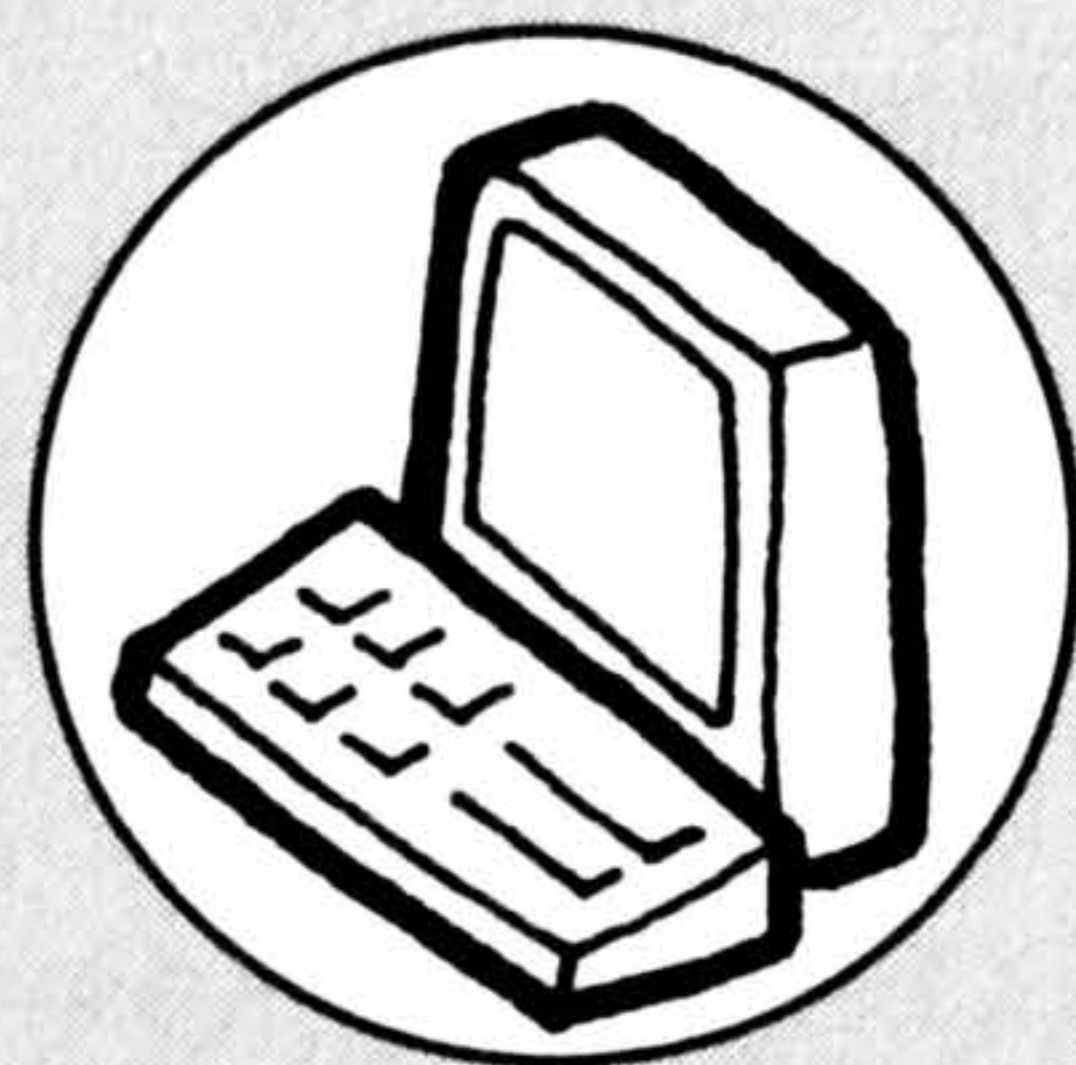
書式例

（1） REN A:TEST1.DAT TEST2.DAT
ドライブAに入っているTEST1というファイル名を、TEST2に変更する。

（2） REN A:TEST1.LST TEST1.PRN
ドライブAに入っているTEST1というファイル名のタイプ名を、LSTからPRNに変更する。

（3） REN A:AB.* AC.*
ドライブBのABというファイルを、すべてACというファイル名に変更する。

注 MSX−DOS命令の中で、FORMAT、DIR、DELETE、ERASE、COPYなどの命令は、一般命令と同じ働きをするため、必要に応じて一般命令としても使える。これらの命令を一般命令として使うとき、MSX−DISK BASICからは「CALL」（省略形として「_」でもよい）を付けて呼び出す。

MSX BASIC
用語用例辞典

❶ダイレクトコマンド

AUTO（オート）

■行番号を自動的に発生させて表示する

プログラムの入力時に行番号を自動的に発生させ、入力待ち状態にする。

書式 AUTO　行番号，増分

説明　行番号はプログラムの文頭の行番号、増分は行番号と次の行の行番号の間隔である。指定した文頭の行番号と指定した増分の行番号間隔が自動的に発生してプログラムにつけられる。行番号を省略すると文頭の行番号は10となる。増分を省略すると行番号の間隔は10となる。

書式例

AUTO	100,20	文頭の行番号100、行番号の間隔は20
AUTO	,5	文頭の行番号10、行番号の間隔は5
AUTO	100	文頭の行番号100、行番号の間隔は10
AUTO		文頭の行番号10、行番号の間隔は10

注

1　行番号はRETURNキーを押すごとに指定増分間隔でつぎつぎに自動的に発生して表示する。

2　動作を中止するときはCTRL+STOPかCTRL+Cキーを押す。そのとき自動発生している行番号は格納されず、BASICコマンドレベルに移行する。

3　行番号の指定範囲は 0～65529 の整数。増分指定範囲は 1～65529の整数。

4　プログラム中すでに使用している行番号が発生した場合、発生した行番号の直後にアスタリスク（*）が表示される。RETURNキーを押せば、古い内容はそのままの状態で残る。新しくキー入力してRETURNキーを押せば、その行は新しい内容に変わる。

5　AUTOモード中スクリーンエディットの機能は動作する。

ＫＥＹ（キー）

■ファンクションキーの内容を定義する

ファンクションキーの内容を新しく定義するときに使用する。

書式　ＫＥＹ　キー番号，"文字列"

説明　キー番号はファンクションキーの番号、"文字列"は与える機能を示す文字列。定義したあとは、そのキーを押せば定義内容のとおりに動作する。

書式例　ＫＥＹ　１，"ＬＯＣＡＴＥ"

１のファンクションキーにＬＯＣＡＴＥを定義した例である。

注

1　ファンクションキーは全部で10個あるから、指定するキー番号は 1～10までである。

2　定義する文字列は最大15文字以内で、コントロール文字も含む。

3　キーボードから入力できない定義文字は、プラス記号を付け、ＣＨＲ＄関数で入力する。

（例）ＫＥＹ　１，"ＣＬＳ"＋ＣＨＲ＄（９）

4　ファンクションキーの内容を調べるときは、ＫＥＹ　ＬＩＳＴとする。

5　いちど定義した内容は、再定義するか電源を切らないかぎり保持される。

KEY LIST（キーリスト）

■ファンクションキーの内容を画面に表示する

10個のファンクションキーに定義されている内容を、画面に表示する。

書式 KEY LIST

説明 KEY LISTを実行すればF1〜F10までの内容が画面にF1, F2…の順で表示される。はじめて電源を入れた時点では、ファンクションキーはつぎのように定義されている。

（画　面）　　　　　　　　　　（説　明）

画面	説明
KEY LIST	ファンクションキー番号
color	……………F1　（⤵はリターンキー）
auto	……………F2
goto	……………F3
list	……………F4
run ⤵	……………F5
color 15, 4, 7 ⤵	……………F6
cload	……………F7
cont ⤵	……………F8
list ⤵	……………F9
cls run⤵	……………F10
ok	
■	

注

1　電源を入れた時点では画面下にF1〜F5までの内容が表示される。SHIFTキーを押せばF6〜F10の内容が表示される。

2　ファンクションキーには15文字までの文字やコントロール文字が定義できるが、画面の下に表示されるファンクションキーの内容は通常5文字まで。5文字以上の定義内容を確認するときはKEY LISTを実行すればいい。

NEW（ニュー）

■メモリからのプログラム消去と変数の初期化

新しいプログラムを入力する前に使用する。今までのプログラムをメモリから消去し、すべての変数をクリアする。

書式 NEW

説明 コマンドレベルでNEWを実行すれば、それまで使用したプログラムをメモリから消去し、初期化する。プログラムの中で使用すれば、実行後はコマンドレベルに戻る。また、開かれたファイルは自動的に閉じられ、DEF FN文の定義は自動的に解消される。

注

1 NEWをコマンドレベルで実行しても、画面に変化は現れない。

2 NEWの実行後、メモリ内のプログラムが消去されたかどうかを確認するためにはLIST命令を実行する。

●サンプルプログラム

```
10 CLS
20 FOR N=1 TO 200
30 PRINT "*"
40 NEXT
50 NEW
```

キー入力後LIST命令でプログラムが入っていることを確認したらRUNしてみる。そのあとでLISTをしてもプログラムは表示されないはずだ。

NEW命令はたいせつなプログラムを一瞬のうちに消去してしまうから、慎重に使用しなければならない。

REM（リマーク）

■プログラム中に注釈を入れる

プログラムの整理のためにプログラム中に適当な注釈文を書く。プログラムを見やすくしたり、手直ししやすくするために使用する。

書式 REM　注釈文

説明 プログラム中のREM文は非実行文とみなされる。プログラムの実行にはまったく影響しないから一連の命令の間以外ならどこにいくつ使用してもさしつかえない。

●サンプルプログラム

```
10　REM　ケイサンルーチン
20　A＝B＊C／2
30　R＝RND（1）＊9：REM　ランダムスウ
```

注

1　プログラムの中で使用するときREMの代わりにアポストロフィ（’）を使用することができる。使用方法はREMと同じである。

2　REM文の中に命令文を入れても注釈文とみなされて非実行文となる。

3　REM文中にコロン（：）を使用しても注釈の一部となるため、マルチステートメントは構成できない。

4　REM文もプログラム一部だから、GOTO文やGOSUB文の飛び先として使用できる。ただし、REM文は非実行文だから、つぎの実行文から実行される。

5　REM文を使用すればプログラム全体が見やすくなるがメモリの領域を使うのでメモリが少なくなるのと、実行速度を下げる結果となるので、最小限におさえた方がいい。

ＲＵＮ（ラン）

■プログラムを実行させる

プログラムの実行をスタートさせるときに使用する。プログラムが実行されると変数の初期化が行われる。

書式 ＲＵＮ　行番号

説明 行番号を指定すればその行から実行される。行番号を指定しなければ、プログラムの最も若い行番号から実行される。プログラムの実行が終わるとコマンドレベルに戻る。

注

1　ＲＵＮ命令はダイレクトモードでもプログラムモードでも使用できる。

●サンプルプログラム

```
10  A=20:B=30
20  A=A+B
30  B=B+A
40  PRINT  A
50  PRINT  B
60  GOTO20      ………RUN
```

このプログラムをＲＵＮ命令でさせてみる。60行のＧＯＴＯ20で20行に飛び、それを何回も繰り返すから、ＰＲＩＮＴされるＡとＢの値は増えつづける。

つぎに60行をＲＵＮに直して実行する。　　　60　ＲＵＮ

実行後のＡとＢの値はいつも一定値である。ダイレクトモードとプログラムモードの使い分けである。

ＬＩＳＴ（リスト）

メモリにあるプログラムを表示する

メモリにあるプログラムを呼び出して画面に表示する。

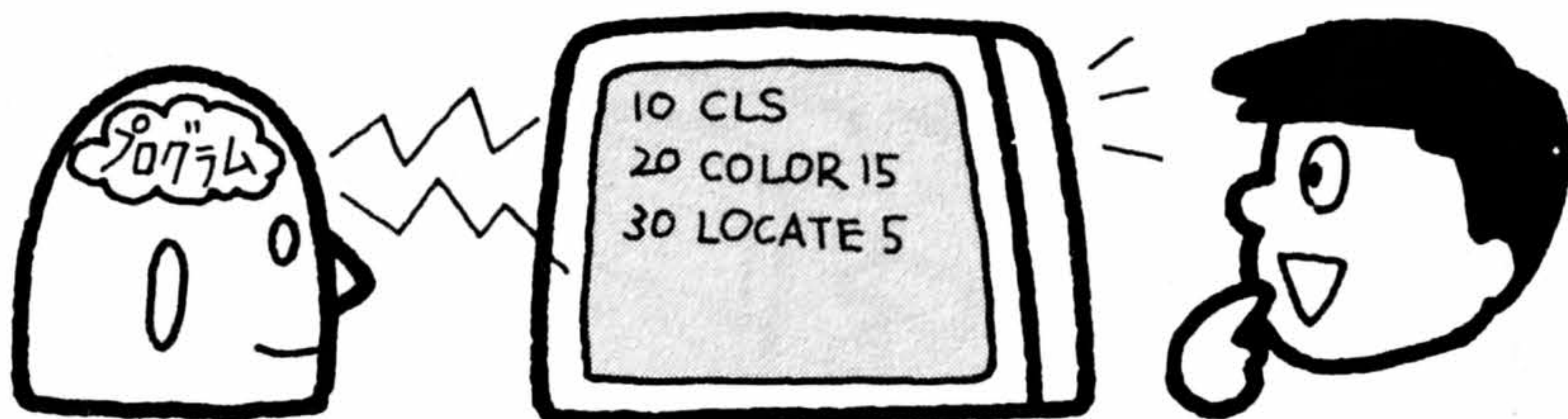
書式　ＬＩＳＴ　行番号①－行番号②

説明　行番号①から行番号②までのプログラムを画面に表示する。行番号①を省略して、－（マイナス）行番号②とすれば、いちばん若い行番号から行番号②までを表示する。行番号②を省略して、行番号①－とすれば、行番号①からプログラムの最後までを表示する。行番号①、行番号②ともに省略すれば、プログラムの最初から最後までを表示する。また、ＬＩＳＴ　行番号とすれば、その行番号のプログラムだけを表示する。

書式例　ＬＩＳＴ……………………プログラムの最初から最後までを表示する。
ＬＩＳＴ　－50…………プログラムの最初から50行までを表示する。
ＬＩＳＴ　50－…………プログラムの50行から最後までを表示する。
ＬＩＳＴ　50－100 ……プログラムの50行から100 行までを表示する。
ＬＩＳＴ　50……………プログラムの50行だけを表示する。

注

1　ＬＩＳＴの途中で ＳＴＯＰ キーを押せば、ＬＩＳＴの動作がそのときの表示のまま中断する。プログラムをゆっくり確認するときなどに使用する。ただし、中断してからプログラムの手直しはできない。もう一度 ＳＴＯＰ キーを押せば、ＬＩＳＴの動作が再開する。

2　ＬＩＳＴの途中で ＣＴＲＬ ＋ ＳＴＯＰ キーを押せば、ＬＩＳＴの動作が終了する。

ＬＬＩＳＴ（エルリスト）

■プログラムをプリンタで印字する

メモリにある行番号つきのプログラムをプリンタに出力して印字する。このコマンドを実行後はコマンドレベルに戻る。

書式　ＬＬＩＳＴ　行番号①－行番号②

説明　行番号①から行番号②までのプログラムをプリンタで印字する。行番号①を省略した場合は、プログラムの最も若い行番号から行番号②までをプリンタで印字する。行番号②を省略した場合は、行番号①からプログラムの最後の行番号までを印字する。行番号①、行番号②ともに省略した場合は、プログラムの最初から最後までをプリンタに印字する。行番号だけを指定した場合は、その行だけ印字する。

書式例　ＬＬＩＳＴ　………プログラムの最初から最後までをプリンタで印字する。

ＬＬＩＳＴ　－20……最初から20行までをプリンタで印字する。

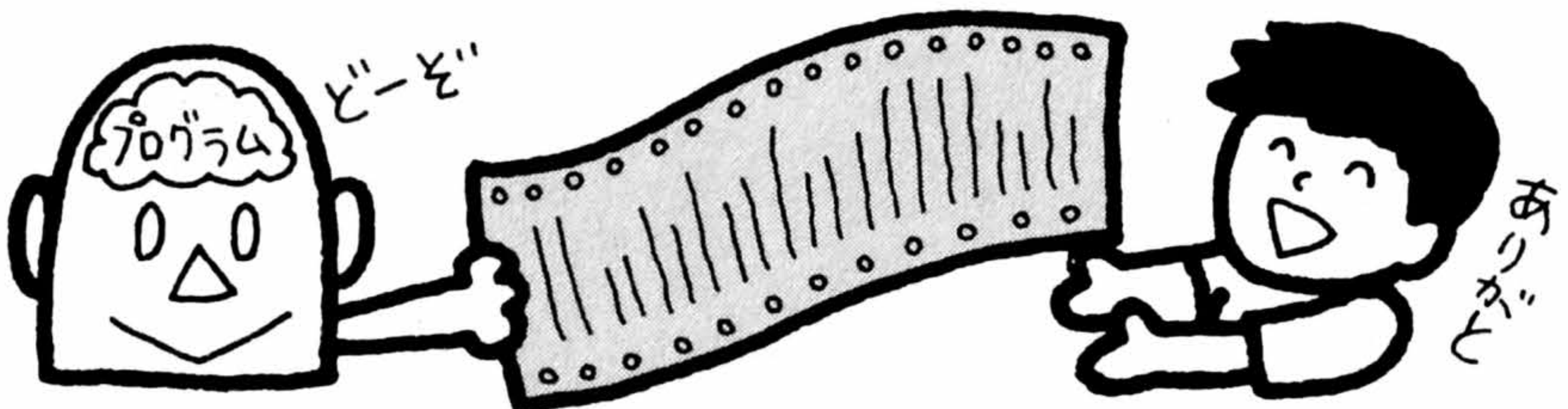
ＬＬＩＳＴ　20－……20行から最後までをプリンタで印字する。

ＬＬＩＳＴ　20－40…20行から40行までをプリンタで印字する。

ＬＬＩＳＴ　20………20行だけをプリンタで印字する。

注

1　プリンタへの出力、印字中に ＣＴＲＬ ＋ ＳＴＯＰ キーを押せば、ＬＬＩＳＴの動作が中断し、コマンドレベルに戻る。

2　プログラムはダブルクォーテーションでくくったもの以外大文字で印字される。

RENUM（リナンバー）

■プログラムの行番号を付け直して整理する

プログラムを作成するとき、行番号を自動的に整理する。行番号の手直しや整理に使用する。

書式 RENUM　新行番号，旧行番号，増分

説明 新行番号は新しくつける行番号の最初の行番号である。旧行番号はつけ替えを始める現在のプログラムの行番号である。増分はこれからつけていく行番号の間隔である。

新行番号を省略すれば、新番号は10から始まる。旧行番号を省略すれば、プログラムの最初の行からつけ替えが始まる。

書式例

RENUM　20，10，5 ……　プログラムの旧行番号10を新番号20として、増分（行間隔）5 で行番号を整理する。

RENUM　20，10　 ……　旧行番号10を20，増分は10で行番号を整理する。

RENUM　，10，10……　プログラムの先頭の行番号を10、増分を10にして行番号を整理する。

RENUM　20,,　10……　プログラムの最初の行番号を20にし、増分を10として行番号を整理する。

RENUM　,,　20……　プログラムの先頭の行番号を10、増分を20で行番号を整理する。

RENUM　100 ……………　プログラムの先頭の行番号を100､増分を10にして行番号を整理する。

RENUM ……………………　プログラムの先頭を10、増分を10にして行番号を整理する。

注

1　プログラムの行番号が新しくなるとともに、GOTO、GOSUBなどで指定する飛び先番号や他の関係行番号も自動的に新しい番号に変更する。し

かし、この命令で指定する行番号が存在しない場合には、Undefined line number というエラーメッセージが表示され、誤った行番号がそのまま画面に表示される。そのときは自分で行番号を訂正する。

2　行番号を新しくすることはできるが、プログラムそのものの順序を変更することはできない。また、行番号は65529 までつけられるが、これ以上の行番号の発生を指名した場合は、Illegal function call というエラーメッセージが画面に表示される。

3　この命令はある行番号からある行番号の間だけの指定はできない。

●サンプルプログラム

```
10  CLS
11  A=5:B=4
15  C=A+B:D=A-B
20  PRINT C
30  PRINT D
35  END
```

このプログラムをRENUMとすると、行番号は下のように変更される。

```
10  CLS
20  A=5:B=4
30  C=A+B:D=A-B
40  PRINT C
50  PRINT D
60  END
```

DELETE（デリート）

■プログラムの一部を削除する

プログラムの中のいらなくなった行をまとめて削除する。

書式 DELETE 行番号①－行番号②

説明 行番号①から行番号②までのプログラムを削除する。行番号①を省略した場合は、プログラムの最も若い行番号から行番号②までを削除する。行番号②を省略した場合は、行番号①からプログラムの最後の行番号までのすべてのプログラムを削除する。行番号を1個だけ指定すれば、指定行番号だけを削除する。

書式例

DELETE 10－100 …… 行番号10行から100 行までのプログラムを削除する。

DELETE 200－ ……… 行番号200 以降のすべてのプログラムを削除する。

DELETE －100 … プログラムの最初の行から100 行までのプログラムを削除する。

DELETE 100 ………… 行番号100 のプログラムを削除する。

注

1 間隔の指定にはマイナスバー（－）を使用する。

2 プログラムの10－100 などのように、ある行からある行までのプログラムを削除する場合には、小さい行番号から大きい行番号の順に指定する。大きい行から小さい行の順に指定するとエラーになる。

3 この命令を使用してプログラムを削除してしまうともとに戻らないから注意する。

TRON/TROFF(トレーサーオン/オフ)

■プログラムの実行行番号を表示する

現在実行しているプログラムの行番号を画面に表示する。

書式 TRON

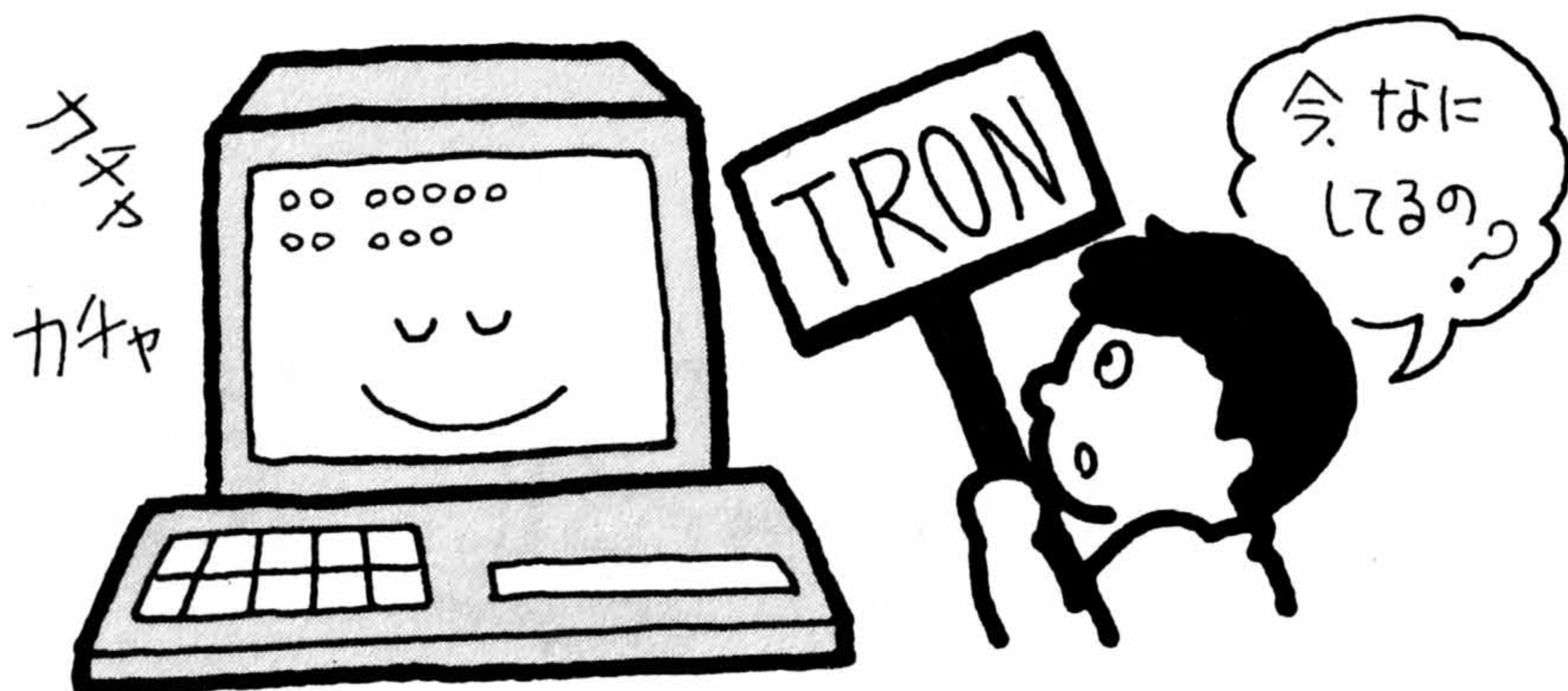
TROFF

説明 電源を入れたときの初期状態はTROFFになっている。トレーサーが必要なときTRONを使用する。TRONの実行は、ダイレクトモードでもプログラムモードでも使用可能である。TRONの実行時、それぞれの行番号は角カッコに囲まれて画面に表示される。

注

1 TRONは、プログラムのデバッグのとき使用すれば非常に便利である。はじめてデバッグするとき、実行状態がわかりにくく、手間どることが多い。そんなとき使用すれば、プログラムの実行状態がわかるから対処しやすい。

2 現在実行中のプログラムの行番号が角カッコで囲まれて表示されるのは、テキストモードのときだけである。グラフィックモードで使用しても画面には表示されない。グラフィックモードでTRONを実行すると、テキストモードに変わったときに表示が始まる。

3 TRONモードは、TROFFを実行するかNEWコマンドを実行すればリセットされる。

ＣＯＮＴ（コンティニュー）

■停止したプログラムの実行を再開する

［ＣＴＲＬ］キーと［ＳＴＯＰ］キーを使ってプログラムを中断したときやＳＴＯＰ文、ＥＮＤ文でプログラムを停止させたとき、プログラムの実行を再開させるために使用する。

書式　［ＣＯＮＴ］

説明　プログラムモードの実行中、ＳＴＯＰ文や［ＳＴＯＰ］キーの入力でプログラムの実行を停止し、ダイレクトモードで変数を調べたり、変数値を変更したりしたあと、プログラムの実行を再開するときに使用する。一時中断や停止したプログラムの次の文から実行が再開する。ただし、ＩＮＰＵＴ文のキー入力で中断したり停止させたときは、つぎの行からではなく、その行の最初から実行が再開される。

注

1　［ＣＴＲＬ］キーと［ＳＴＯＰ］キーを使ったり、ＳＴＯＰ文やＥＮＤ文などでプログラムの実行を中断し、コマンドレベルになっているときにはＬＩＳＴやＰＲＩＮＴ等のコマンドをダイレクトに実行することはできるが、プログラムの内容を変更する場合は、再実行命令であるＣＯＮＴ文の使用はできなくなる。

　また、プリンタに出力中に中断した場合もＣＯＮＴ文は使用できなくなる。

2　ＣＯＮＴ文はプログラムのデバッグには欠かせない命令である。プログラム実行中に手直ししたいときなど、プログラムにＳＴＯＰ文を入れておき、プログラムを中断させて、変数代入値等を調べ、プログラムのデバッグに役立てる。その後、ＣＯＮＴ文を実行させる。

CSAVE（カセットセーブ）

■メモリのプログラムをテープに記録保存

メモリに記憶させたプログラムは、電源をOFFにしたり、NEW文を使用したりすると、メモリから消えてしまう。プロクラムを保存するために、バイナリ形式でカセットに記録する。

書式 CSAVE "ファイル名" ，ボーレート

説明 プログラムを記録保存する場合、ファイル名をダブルクォーテーションで囲んで指定する。ファイル名は最大6文字までの文字列である。ボーレート（転送速度）は、1200ボーと2400ボーが指定できる。ボーレートの初期値は1200ボーになっている。1200ボーは1，2400ボーは2で指定する。

書式例 CSAVE "MSX" …………ファイル名をMSXとしてテープにセーブし、このときのボーレートは1200ボーである。

注

1 セーブ時のファイル名は最大6文字以内の文字列で指定する。7文字以上で指定してもかまわないが、先頭から6文字までが有効になり、以降は無視される。

2 ボーレートはカセットテープへ記録するときの転送速度である。ボーレートを省略した場合は、SCREEN文で指定したボーレートが採用されるが、初期値は1200ボーになっている。また、2400ボーでの記録は、テープの種類によってはセーブミスを生ずる場合がある。

3 CSAVEが終わったら、うまく記録されているかどうかを "CLOAD ?" で確認しておく。ミスがなければ画面にOKマークが表示される。

CLOAD／CLOAD？(カセットロード)

■テープからプログラムを読み込む

カセットテープに記録されているプログラムを本体に読み込む。

書式　CLOAD　"ファイル名"

　　　　CLOAD？　"ファイル名"

説明　カセットテープに記録されているプログラムのファイル名を指定して本体に読み込む。ファイル名は６文字以内で指定する。ファイル名を省略した場合は、最初に見つけたファイル名をFound と表示しＬＯＡＤする。ＣＬＯＡＤ？はプログラムをセーブしたときの確認コマンドである。

書式例　ＣＬＯＡＤ…………………　最初に見つけたプログラムを読み込む。

　ＣＬＯＡＤ "ＭＳＸ" ……　ファイル名ＭＳＸのプログラムを読み込む。

　ＣＬＯＡＤ？………………　最初に見つけたプログラムを確認する。

　ＣＬＯＡＤ？ "ＭＳＸ" …　ファイル名ＭＳＸのプログラムを確認する。

注

1　ＬＯＡＤ時のファイル名は６文字以内で指定するが、７文字以上の場合は先頭の６文字までが有効である。ファイル名を見つけるとＦｏｕｎｄ：（ファイル名）と画面表示されて、プログラムのロードが開始される。

　また、ＬＯＡＤ中に別のファイル名を見つけた場合は、Ｓｋｉｐ：（ファイル名）と画面表示され、再度目的のプログラムをさがす。

2　ＣＬＯＡＤ？は、ＳＡＶＥ命令でプログラムを記録した後に、間違いなくＳＡＶＥされているかどうかを確認するためのコマンドである。プログラムをテープに記録した場合はかならずＣＬＯＡＤ？で確認する。

プログラム
どーぞ

❷画面表示コマンド

PRINT（プリント）

■テキスト画面に各記号を表示させる

画面に英数字や記号等を表示したいときに使用する。その表示位置は前もって指定された場所からである。

書式 PRINT "文字列" または "数式" または "変数" または "数値"

説明 PRINT命令のあとには、表示させたい数式、変数、数値やダブルクォーテーションで囲んだ文字列などを指定する。PRINTだけで、あとを省略した場合は、改行だけが実行される。

「PRINT」を「LPRINT」とすると、画面ではなくプリンタに出力される。

書式例

PRINT "MSX"……	画面にMSXを表示する。
PRINT A+B…………	変数AとBの内容をたした数値を画面表示する。
PRINT 10*2+6……	計算の結果を画面に表示する。
PRINT 180 ……………	画面に180 を表示する。
PRINT……………………	改行だけを実行する。
LPRINT…………………	プリンタに出力する。

注

1 PRINT文は "?"（クエッションマーク）だけで代用できる。

```
10 ? A+B
20 ? "MSX"
```

2 PRINT文で表示される数値の前には＋記号か－記号が表示できるようにスペースがあいている。数値が正の数であれば＋は省略されて空白になり、負の数であれば－が表示される。

3　ＰＲＩＮＴ文の最後にコンマ（，）やセミコロン（；）を付ければ文をつなぐことができる。コンマは13字間隔をとり、セミコロンは、つぎのＰＲＩＮＴ文の内容を続いて表示する。

●サンプルプログラム

```
100 CLS
105 COLOR 3
110 PRINT" #   # "
120 PRINT"# # # #"
130 PRINT"#  #  #"
140 PRINT" #   # "
150 PRINT"  # #  "
160 PRINT"   #   "
170 PRINT"  $$$  "
180 PRINT"$$   $$"
190 PRINT"  ﾊｰﾄ  "
200 PRINT"$$$$$$$"
```

PRINT USING（プリント ユージング）

■表示する書式を指定する

画面に表示させる文字や数値の書式を指定する。テキスト画面の編集に使用すると便利である。

書式　ＰＲＩＮＴ　ＵＳＩＮＧ“制御文字”；文字列または変数または数値

説明　文字列や変数や数値は複数で指定してもいい。この場合は指定された書式に従って、テキスト画面に表示する。表示させるコンマ（,）かセミコロン（；）で区切る。特定のキャラクタ（制御文字）を使用して文字列の編集を行なったり、変数や数値を編集したりする。

●変数や数値を編集するための制御文字

1　＃（ナンバーサイン）

制御文字＃で指定した桁数だけ、変数や数値を画面に表示する。表示する数値が指定した桁数より少なければ、右づめで画面に表示され、左側は空白になる。また、指定した桁数より多ければ左側に“％”が表示される。

数値に小数点がついている場合には小数点以下の数値は無視される。

```
10 CLS
20 A=20：B=30
30 PRINT USING"####"；A，B
```

```
  20  30
OK
■
```

```
40 END
```

2 ・(小数点)

小数点の位置を指定するが、小数点は制御文字＃の中で使用される。ただし小数点以下が指定した桁数より小さい場合は、右側から自動的に0が補われる。

```
10 CLS
20 PRINT USING "##.##" ;20, 30.5
```

```
20.00  30.50
OK
■
```

3 ＋・－(プラス・マイナス)

制御文字＃の前後に＋または－の記号をつければ、数値の前に＋または－の記号が表示される。ただし、＋または－の記号を2個以上つけると、外側の記号は、制御文字と判断されない。

```
10 CLS
20 PRINT USING "+##.##" 5, 3.5
```

```
+5.00  +3.50
OK
■
```

4 ＊＊(アスタリスク)

制御文字＃の前に＊＊をつければ、数値の整数部の桁数が指定した桁数より少ない場合には、数値の左側に＊が表示される。ただし、指定した桁数とは、＃と＊の合計の桁数である。

5 ￥￥(エンサイン)

制御文字＃の前に￥￥をつければ、数値の前に￥を表示する。表示する数値の桁数は、＃と￥を加算したものから1だけ減じたものになる。表示桁数が指定した桁数より多い場合、￥マークの前に％の表示が入る。ただし、指定数式の書式指定をしているときは、￥￥は使用できない。

6 ,(コンマ)

制御文字#の並びの中で“,”を指定すれば、数値の整数部が最小桁から3桁ずつ“,”で区切られて表示される。

```
10 CLS
20 PRINT USING"#####,.#";1234.56
```

```
 1,234.56
OK
■
```

7　制御文字以外の文字

書式内に制御文字以外の文字（英字、数字、カナ、グラフィック記号等）を入れた場合、数値の前後で文字に相当する位置に指定した文字が表示される。

●文字列を編集するための制御文字

1　!

書式に“!”を使用すれば、文字列の最初の1字だけが表示される。

PRINT USING"!";"ABCD"………Aだけが表示

2　&　&

&と&で囲まれた空白の数に2（&&に相当する）を加えた数の文字が、文字列の左側から選ばれて表示される。文字列の文字が指定した文字数より少ない場合、文字列は左づめで表示され、残った部分は空白となる。

PRINT USING"&　　　&";"MSX12345"

………MSX123と表示

3　@

書式中に@を使用すれば、@の場所に文字列が代入される。複数個指定した場合は、@1個が文字列1個に相当する。

PRINT USING"This is a @.";"MSX"

………This is a MSX. と表示

●サンプルプログラム

```
10 CLS
20 FOR N=1TO10
30 LOCATE4,N
40 PRINTUSING"#.####";5/N
50 LOCATE20,N
60 PRINTUSING"#.##";5/N
70 NEXT
```

CLS（クリアスクリーン）

■画面の表示を消去する

画面に表示されている文字や数字、記号、グラフィックなどをすべて消す。

書式 CLS

説明 この命令は、テキストモードとグラフィックモードともに使用できる。テキストモードでCLSを使用した場合、カーソルは画面左上の位置、X座標0，Y座標0，すなわち（0，0）の位置に移動する。

ただし、画面にファンクションキーのリストが表示されている場合は、CLSを実行してもその表示は消去されない。

注

1 CLSを実行しても画面を消去するだけで、各変数の値には影響しない。

2 COLOR命令で、背景色を変えたい場合は、必ずこのCLS命令を実行してから背景色を指定する。

●サンプルプログラム

```
10 CLS
20 COLOR 3,1,9
30 PRINT:PRINT"    コノ ケイサン ヲ シマショウ"
40 A=INT(RND(-TIME)*9+1)
50 B=INT(RND(-TIME)*9+1)
60 PRINT:PRINT:PRINT"     ";A;"+";B;"="
70 LOCATE14,4:INPUT C
80 IFC=A+BTHEN100ELSE120
90 END
100 LOCATE 10,10
110 PRINT"アタリ":FOR N=1TO500:NEXT:CLS:GOTO30
120 LOCATE 10,10
130 PRINT"ハズレ":FOR N=1TO500:NEXT:CLS:GOTO30
```

ＬＯＣＡＴＥ（ロケート）

■テキスト画面のカーソル位置を指定する

テキスト画面で、ＰＲＩＮＴ文などを使用して文字や記号を表示させるとき、カーソルの位置を指定する。

書式　ＬＯＣＡＴＥ　Ｘ座票，Ｙ座票，カーソルスイッチ

説明　カーソルをＸ座標とＹ座標で指定したテキスト画面の位置に移動させる命令である。ＰＲＩＮＴ文やＩＮＰＵＴ文の補助的な役割として使用し、表示位置や入力位置を移動させる。Ｘを０，Ｙを０に指定すると画面左上の位置になる。

カーソルスイッチはカーソルの表示状態を０か１で指定する。

０：カーソル表示はされない。

１：カーソルが表示される。

カーソルスイッチの指定は省略できる。また、初期値は０である。

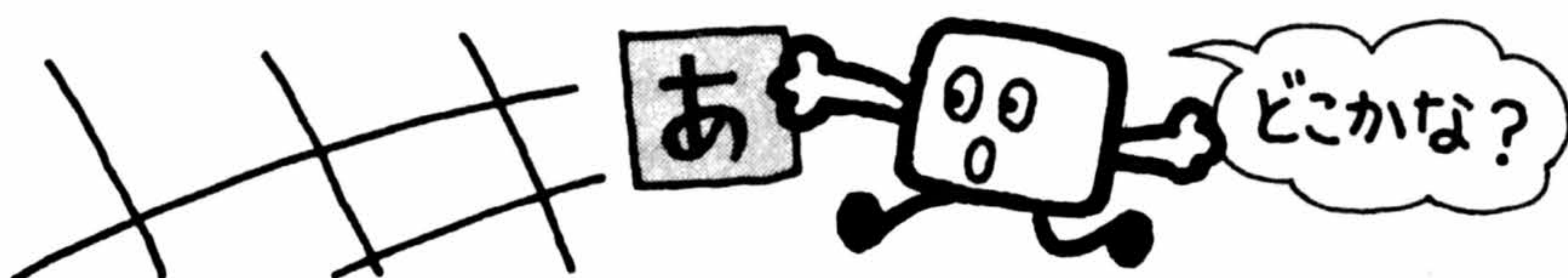
なお、ＬＯＣＡＴＥ文は、ＳＣＲＥＥＮ　０と１のテキストモードだけで使用し、２と３のグラフィックモードでは使用できない。

書式例　ＬＯＣＡＴＥ　Ｘ………………Ｙ座標を省略すると、現在位置のＹの値がそのまま使用される。

ＬＯＣＡＴＥ　，Ｙ……………Ｘ座標を省略すると、現在位置のＸの値がそのまま使用される。

ＬＯＣＡＴＥ　Ｘ，Ｙ…………座標（Ｘ，Ｙ）にカーソルが移動する。

ＬＯＣＡＴＥ　Ｘ，Ｙ，０……座標（Ｘ，Ｙ）の位置にカーソルがあるが、表示はされない。

ＬＯＣＡＴＥ　Ｘ，Ｙ，１……座標（Ｘ，Ｙ）の位置にカーソルが表示される。

```
10 CLS
20 LOCATE 1, 2
30 PRINT "M"
40 LOCATE 2, 3
50 PRINT "S"
60 LOCATE 3, 4
70 PRINT "X"
80 END
```

	0	1	2	3	4	5	6
0							
1							
2		M					
3			S				
4				X			
5	O	K					
6	■						

電源投入時の画面表示領域の初期設定はＷＩＤＴＨ２９（29×24）の状態になっている。Ｘ座標は29桁（0～28）、Ｙ座標は24行（0～23）である。最大は32×24までＷＩＤＴＨ文で指定できるが、このときのＸ座標は32桁（0～31）になる。したがって、ＬＯＣＡＴＥ文で指定する座標の最大値はＷＩＤＴＨ文で設定したものに従わなければならない。

ＳＣＲＥＥＮ　0のときは同じように、Ｘ座標は40桁（0～39）、Ｙ座標は24行（0～23）である。

注

1　ＬＯＣＡＴＥ文は、表示位置や入力位置を指定するだけで、表示させる命令ではない。したがって、ＬＯＣＡＴＥ文のあとにＰＲＩＮＴ文やＩＮＰＵＴ文を書き、ＬＯＣＡＴＥ文で指定してから表示させる。

2　カーソルスイッチを表示させるかどうかの指定は、プログラムモードでもいい。ダイレクトモードのときは、ＬＯＣＡＴＥ　，　，0のようにカーソルの座標位置指定を省略して、カーソルスイッチを切り換えることができる。

●サンプルプログラム

```
10 CLS
20 KEY OFF
30 COLOR 4,1,9
40 FOR N=1 TO 10
50 LOCATE 5+N,5:PRINT "*"
60 NEXT
70 FOR N=1 TO 9
80 LOCATE15-N,5+N:PRINT"*"
90 NEXT
100 FOR N=1 TO 10
110 LOCATE 5+N,15:PRINT"*"
120 NEXT
130 LOCATE9 ,9:PRINT"*"
140 LOCATE11,11:PRINT"*"
150 LOCATE0,20
```

WIDTH（ウィドス）

■画面に表示させる桁数を指定する

ファンクションキーのリスト以外の画面表示を消去して、1行に表示できる任意の桁数を指定する。ただし、8以下の桁数を指定するとファンクションキーのリストは消える。

書式 WIDTH　数値または変数

説明 WIDTHのあとにくる数値あるいは変数で画面に表示する1行の桁数を変えることができる。電源投入時は、X座標が29、Y座標が24（29×24）に指定されている。これをWIDTH命令で最小1から最大32まで指定することができる。

テキストモードの80×24で、X座標は1～80までが指定できる。また、32×24のテキストモードで、1～32まで指定できる。

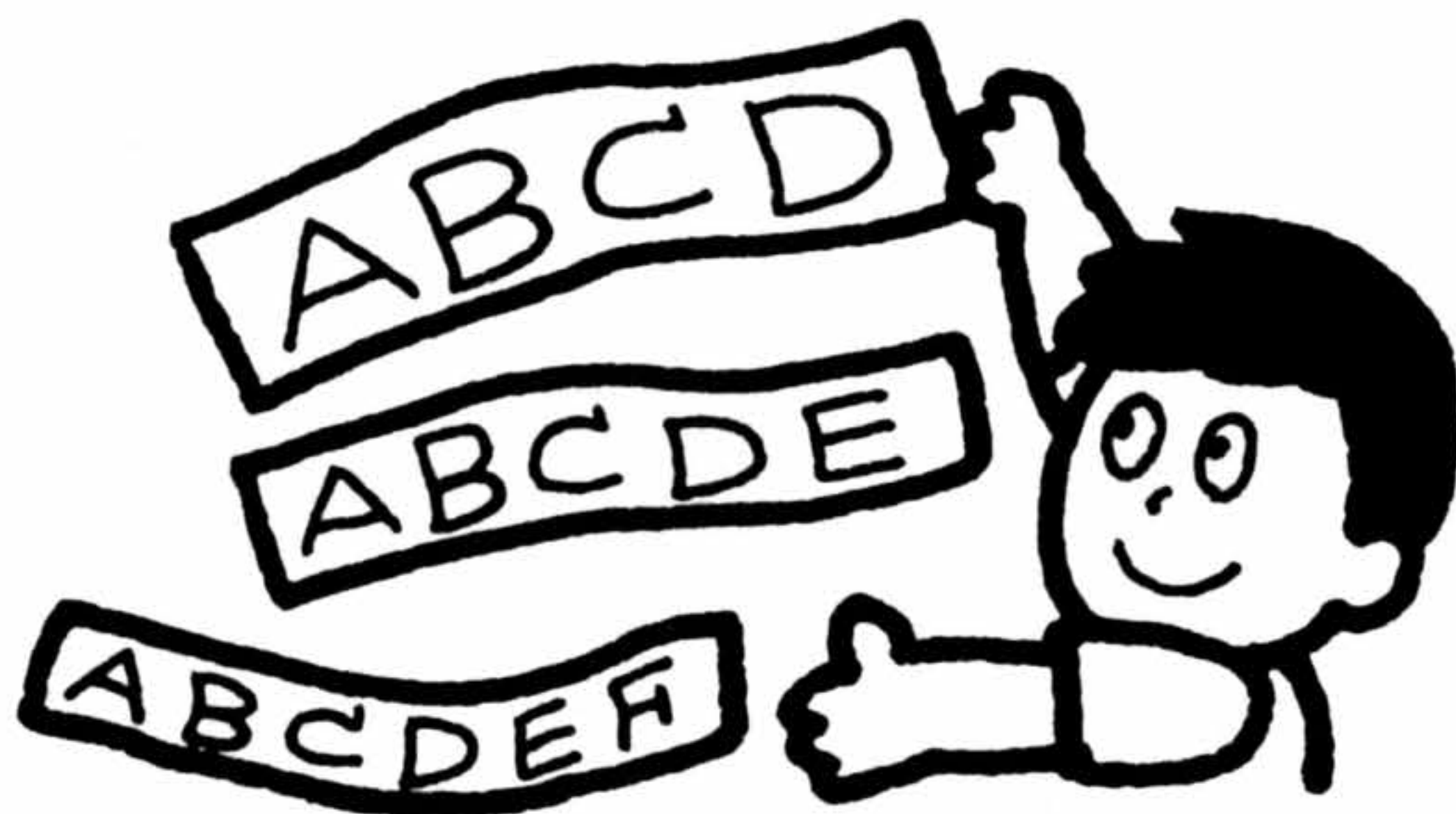
書式例 WIDTH　20………　X座標を20文字モードに指定する。
WIDTH　A………　X座標をAの値の範囲モードに指定する。

注

1　WIDTH命令で指定した範囲をプログラムで再実行した場合、画面全体はクリアされる。

2　WIDTH　1とすると画面の中央の1字が表示される。

●サンプルプログラム

```
100 CLS
110 WIDTH 20
120 PRINT"******"
130 PRINT"  **  "
140 PRINT"  **  "
150 PRINT"  **  "
160 PRINT" **** "
170 PRINT"******"
180 PRINT"####"
```

COLOR（カラー）

■画面の各部分の色を指定する

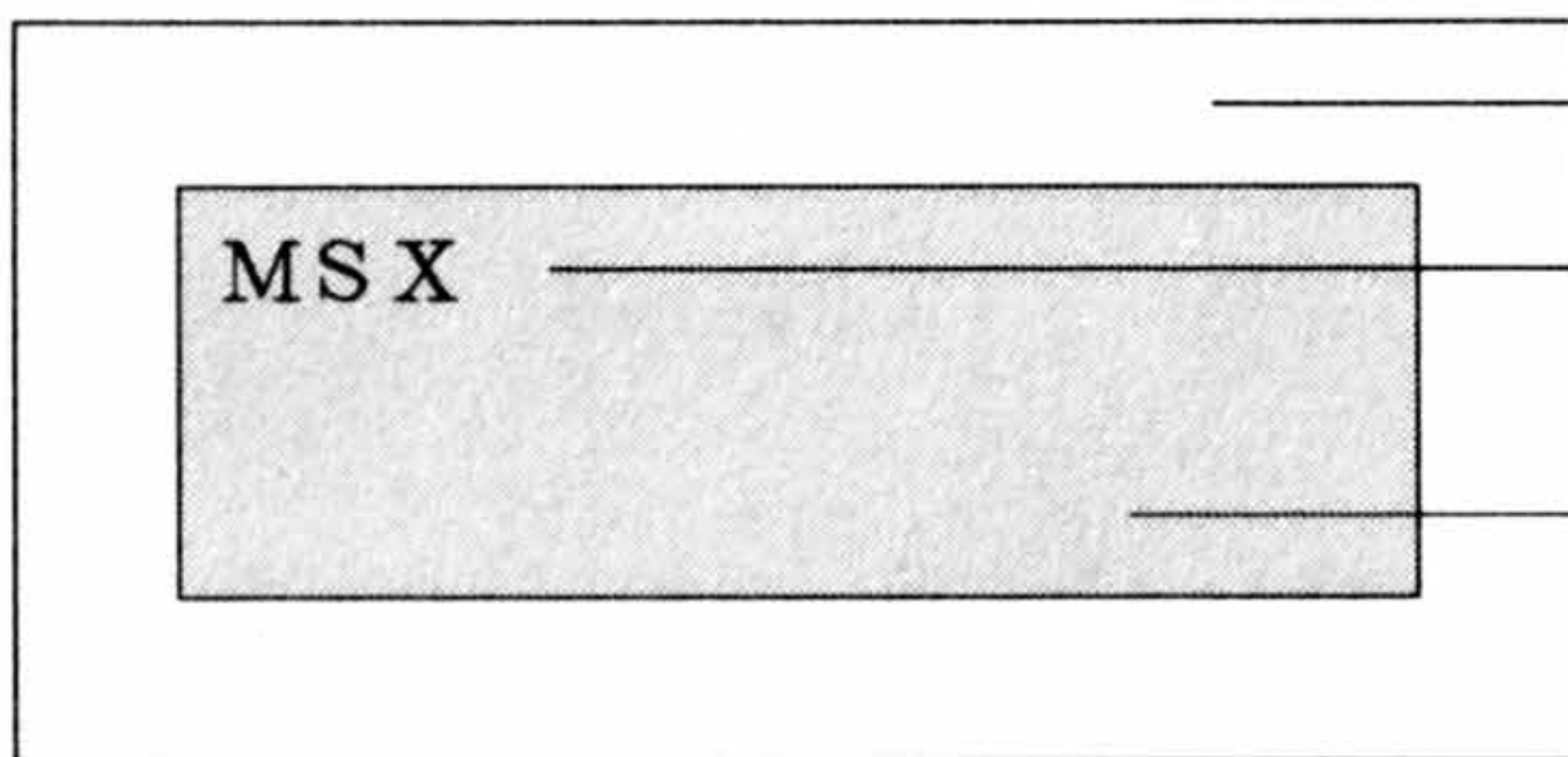
画面の表示色、背景色、周辺色をカラーコード表の数値を使って指定する。

書式 COLOR　表示色，背景色，周辺色

説明 画面の表示色、背景色、周辺色をそれぞれのカラーコード（0～15）で指定する。COLOR文の色指定は、表示色、背景色、周辺色のうち、変える必要のないものを省略することができる。省略すれば、COLOR文を実行する前の色が指定される。

電源を入れたときのカラーコードの初期値は、表示色が白（15）、背景色が暗い青（4）、周辺色が水色（7）になっている。また、このカラーコードの初期値はファンクションキーのＦ６に入っている。Ｆ６のキーを押せば、電源起動時の色となる。

表示色、背景色、周辺色は次の通りである。

MSX

周辺色（BASIC で使用できない領域部分の色）

表示色（英数字や各種グラフィックの色）

背景色（BASIC で使用できる領域部分のバックの色）

●カラーコード表

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
周辺色と同色	黒	緑	明るい緑	暗い青	明るい青	暗い赤	水色	赤	明るい赤	暗い黄色	明るい黄色	暗い緑	紫	灰色	白

書式例　COLOR　1，2，15……　表示色を黒、背景色を緑、周辺色を白に指定する。

COLOR　，1，2　……　背景色を黒、周辺色を緑に指定する。表示色の指定は省略されている。

COLOR　2　……　表示色を緑に指定する。背景色と周辺色の指定は省略されている。

（例）

```
10 CLS
20 FOR I=0 TO 15
30 FOR J=0 TO 15
40 COLOR I,J
50 LOCATE 10,10
60 PRINT "MSX"
70 FOR N=0 TO 100 :NEXT N
80 NEXT J,I
90 END
```

上のプログラムを実行すると、画面に表示されている文字が15色表示されると背景色が変わり、表示色が白色になるとプログラムは終わる。

注

1　テキストモードのとき、背景色はCOLOR文実行後すぐに指定色になる。グラフィックモードのときは、COLOR文実行後、CLS文を実行して画面を消去すると、指定した背景色に塗り変えられる。また、COLOR文実行後，PRESET命令を無指定で実行しても、この背景色が採用される。

■512色のカラーを指定する

従来のMSXの表示色が最高16色までだったのに対して、MSX2では、赤、緑、青の3色の色の濃さを調節して組み合わせることによって、最高512色までを表示させることができる。

書式 COLOR=（パレット番号，赤色数値，緑色数値，青色数値）

説明 各色の数値は、0～7の範囲で指定する。この数値は、各色の濃さを表わし、その数値の変化で512色を表示させる。

書式例 COLOR=（5，7，7，0）

5のパレット番号に、赤色を7、緑色を7、青色を0で指定する。ある濃さ（この場合は7）の赤色と緑色の絵の具を混ぜ合わせたと考えればいい。表示される色は、明るく美しい黄色である。

注

1　COLOR命令で、パレット番号に色を指定したときに、別のプログラムを実行すると、前に指定した色がそのまま表示されることがある。このようなときには、パレットを初期化しておく。初期化の書式は、次のとおりである。

COLOR=NEW

この命令を、プログラム中の色指定を行なう前に置いて初期化する。このほかに、一度電源を切って初期化する方法もある。

COLOR SPRITE(カラースプライト)

■スプライトパターンの横1列ごとに色をつける

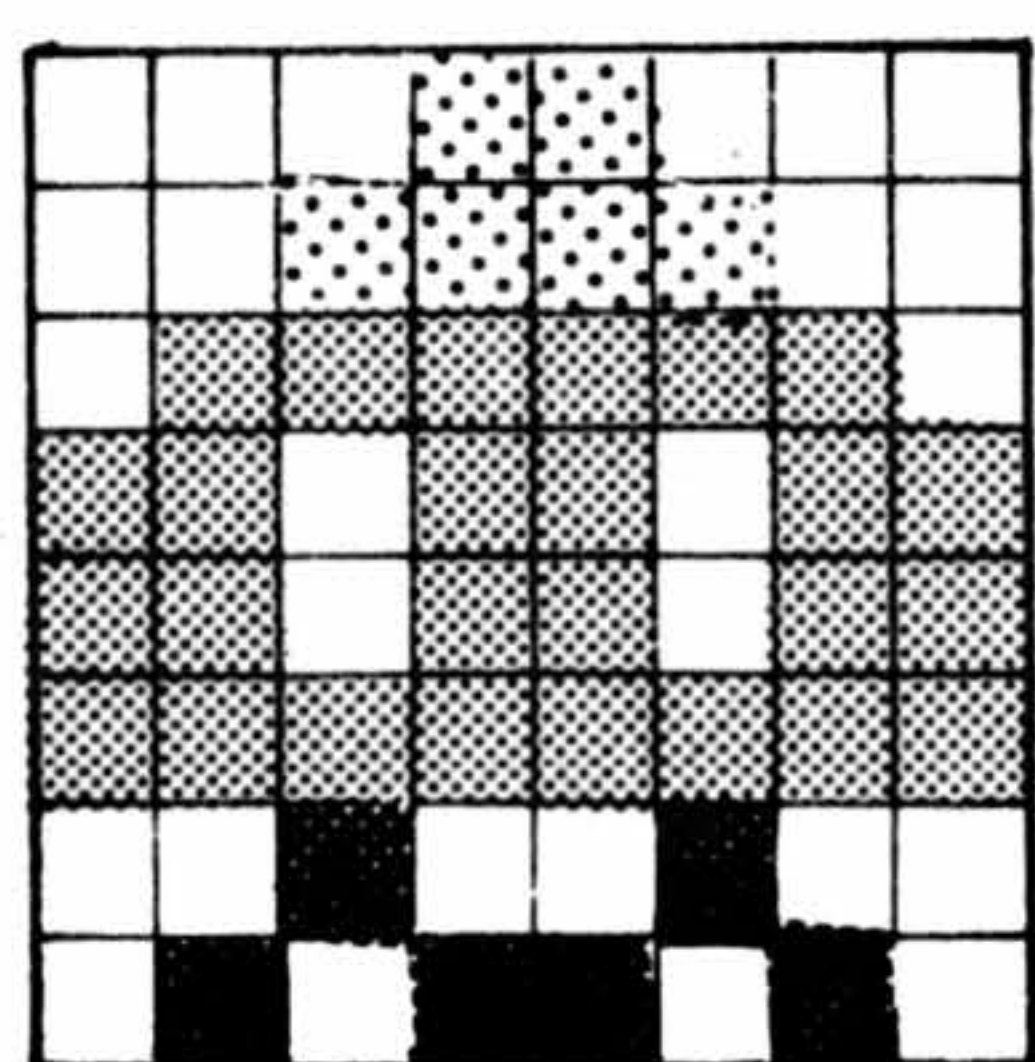
書式 COLOR　SPRITE$（スプライト番号）=CHR$（n）+CHR$（n）+……………

説明 スプライトパターンの横1列ごとに色を指定して表示する。

CHR$（パターン番号）を「+」（プラス記号）でつないで、色を指定する。8×8のスプライトパターンであれば、CHR$（n）が8個並ぶことになり、16×16であればCHR$（n）が32個並ぶことになる。色の指定を省略したと

きの列の色は、前に実行したパターンの色指定がそのまま残る。

ＳＣＲＥＥＮ　8の画面モードでは、同時に256色までを使うことができる。この場合は、パレット番号で指定するのではなく、色ごとにそれぞれ決まっているカラーコード番号で指定する。

書式例　ＣＯＬＯＲ　ＳＰＲＩＴＥ＄（１）＝ＣＨＲ＄（３）＋ＣＨＲ＄（３）＋ＣＨＲ＄（４）＋ＣＨＲ＄（４）………ＣＨＲ＄（２）

ＣＨＲ＄(３)
⋮
ＣＨＲ＄(４)
⋮
＋
ＣＨＲ＄(２)

注

１　ＣＯＬＯＲ　ＳＰＲＩＴＥのあとに、ＰＵＴ　ＳＰＲＩＴＥ命令で色を指定すると、ＣＯＬＯＲ　ＳＰＲＩＴＥで指定した色が無効になってしまうので注意する。このようなときは、もう一度、ＣＯＬＯＲ　ＳＰＲＩＴＥ命令で色を指定するか、または、ＰＵＴ　ＳＰＲＩＴＥで色を指定しない、あるいは省略したほうがいい。

SPC（スペース）

■空白を表示させる

画面に表示したりプリンタで印字したりするとき、任意の長さの空白を作って表示させる。

書式　ＳＰＣ　数値（または変数、または数式）

説明　ＳＰＣ命令は、ＰＲＩＮＴ文やＬＰＲＩＮＴ文など、表示や出力文とともに使用する。このとき指定できる数値は 0～255 の範囲であり、指定した数値分の空白が表示される。

書式例　ＰＲＩＮＴ　ＳＰＣ（4）；"ＭＳＸ"

上の例では、画面にスペースを４個分あけてＭＳＸと表示する。

（例）10　CLS
　　　20　PRINT　"MSX"；
　　　30　PRINT　SPC（4）；"デス"
　　　40　END

```
  012345678
0 MSX    デス
1 OK
2 ■
```

注

1　TAB命令は左側から空白を表示するのに対して、SPC命令は前にある文字列で最後から数えて、その数値分の空白をあける。前になにもない場合はTABと同様である。

●サンプルプログラム

```
100 CLS
110 COLOR13,1,4
120 LOCATE7,4:PRINT"ﾒﾆｭｰ":PRINT:PRINT:PRINT
130 READ A$,B
140 IF B=0 THEN 180
150 A=LEN(A$)
160 PRINTA$;SPC(15-A);"¥";B
170 GOTO130
180 END
190 PRINTA$;SPC(15-A);"¥";B
200 GOTO160
210 END
220 DATA ｺｰﾋｰ,350,ﾚﾓﾝﾃｨｰ,300
230 DATA ﾐﾙｸ,280,ｼﾞｭｰｽ,420
240 DATA ｻﾝﾄﾞｲｯﾁ,450,ｽﾊﾟｹﾞｯﾃｨ,520
250 DATA ﾋﾞｰﾌｶﾚｰ,550,end,0
260 DATA ｺｰﾋｰ,350,ﾚﾓﾝﾃｨｰ,300
270 DATA ﾐﾙｸ,280,ｼﾞｭｰｽ,420
280 DATA ｻﾝﾄﾞｲｯﾁ,450,ｽﾊﾟｹﾞｯﾃｨ,520
290 DATA ﾋﾞｰﾌｶﾚｰ,550,end,0
```

TAB（タブ）

■指定した桁まで空白を表示させる

現在カーソルのある行の任意の位置まで空白を出力し、カーソルを指定位置まで移動させる。

書式 TAB　（数式）

説明　あける空白を指定する数式は、数値でも変数でもよい。PRINT文やLPRINT文中で使用する。使用できる数値は 0〜255 の範囲である。

書式例　TAB（6）……　画面左から6桁分空白を作り、その位置にカーソルが移動する。

（例）
```
10　CLS
20　PRINT　TAB（4）；"MSX"
30　END
```

	0	1	2	3	4	5	6
0					M	S	X
1	O	K					
2	■						
3							

上のプログラムを実行すると、CLS命令でカーソルの座標位置は（0，0）となる。そこでTAB（4）を実行すればY座標はそのままで、X座標の値は4となり、その位置からMSXと表示される。

●サンプルプログラム

```
100 CLS
110 FORN=1TO696
120 PRINT"#";
130 NEXT
140 LOCATE0,5
150 PRINTTAB(10);"$$$"
160 PRINTTAB(10);"$$$"
170 LOCATE0,10
180 PRINTTAB(20);"%%%"
190 PRINTTAB(20);"%%%"
200 PRINTTAB(15);"MSX"
```

KEY ON/OFF(キー オン/オフ)

■ファンクションキー内容を表示させるかさせないか

ファンクションキーの内容を画面の一番下の行に表示させるかさせないかを指定する。

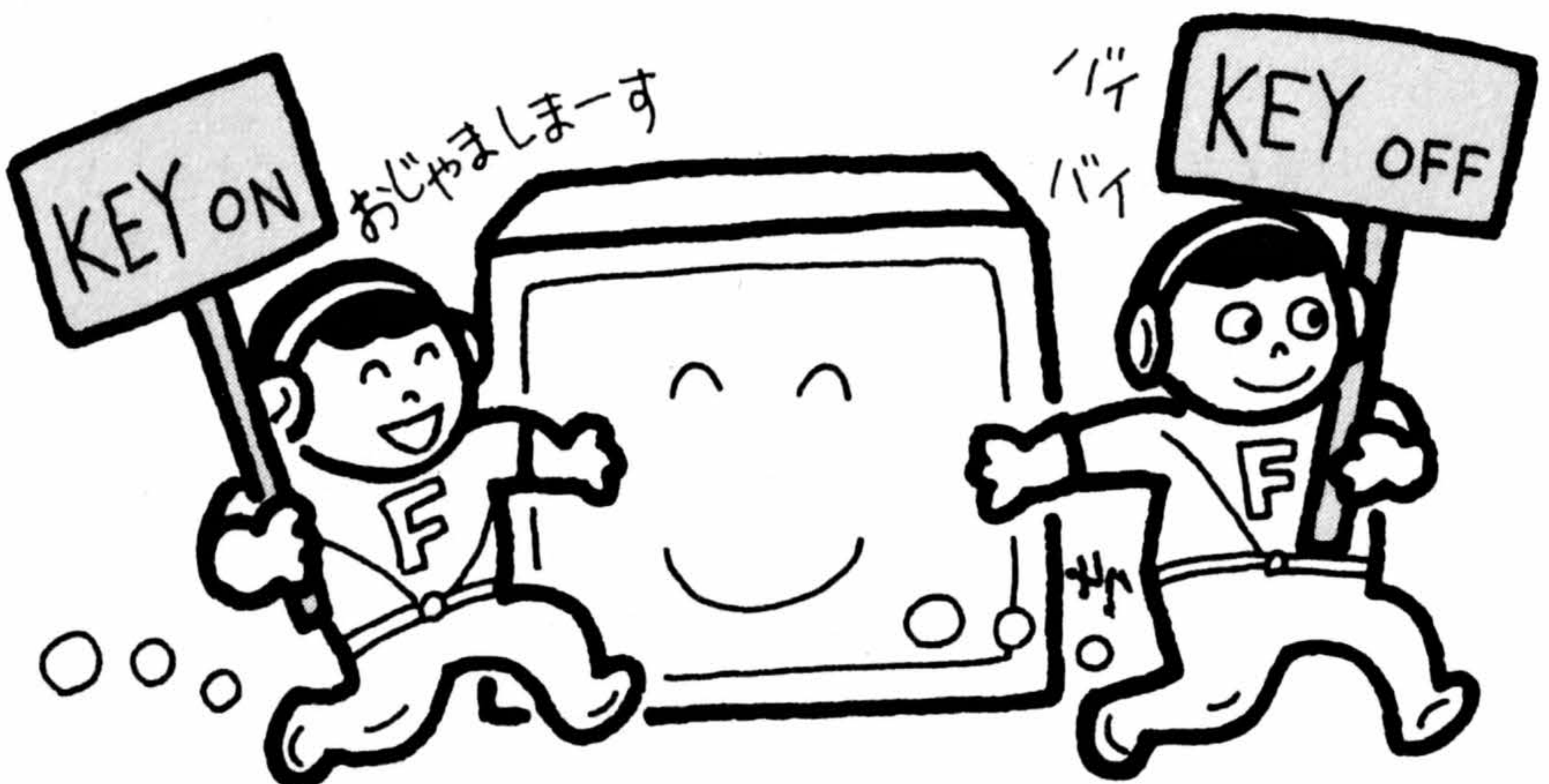
書式 KEY ON　　KEY OFF

説明 ファンクションキーの内容は画面最下位行に表示されるが、この表示を消すときにKEY OFFとし、消した表示を復活表示させるときKEY ONとする。通常表示されるファンクションキーの内容はF１～F５までである。F６～F７までの内容を表示させる場合はSHIFTキーを押す。

注

1 ファンクションキーの内容は、テキストモードのときだけ表示される。グラフィックモードでは表示しない。

2 KEY ON状態でファンクションキーの内容が表示されているときは、画面に表示できる行数が１行分減る。

3 WIDTH文で少ない表示桁を指定すれば、それだけファンクションキーの表示数は減る。WIDTH文指定値が８以下になると、ファンクションキーの内容は表示されない。

SCREEN（スクリーン）

■画面のモードを指定する

MSXの持つ9種類の画面モードを指定するほか、スプライトパターンのサイズ、キークリックスイッチ、カセットのボーレート、プリンタのモードをそれぞれ指定する。

書式

SCREEN　画面モード，スプライトサイズ，キークリックスイッチ，カセットボーレート，プリンタモード

●画面モードの指定

9種類の画面モードを0～8の数値で指定する。この命令の各パラメータはそれぞれ省略して、必要なものだけ指定することができるが、いずれか1以上のパラメータは必ず指定しなければならない。なにも指定せず、ただSCREENとすると"Missing Operand の表示が出る。

画面モードの書式例

SCREEN　n	nは0～8の数値である。各数値ごとにつぎのような指定となる。
SCREEN　0	最大80桁×24行を表示するテキストモードを指定する。ただし通常は79×24である。80×24を表示させるには、別にWIDTH　80の命令を実行させる。
SCREEN　1	最大32桁×24行を表示するテキストモードを指定する。ただし通常は29×24である。32×24を表示させるには、WIDTH　32を実行させる。
SCREEN　2	256 ×192 ドットの高解像度グラフィックモードを指定する。この画面モードでは、点を付けたり消したりして絵を描くことができる。ただし、家庭用テレビをディスプレイに使用した場合、画面の端が切れて見えなくな

	る。画面全体では49152ドットである。
SCREEN　3	64×48ドットのグラフィック、マルチカラーモードを指定する。この命令を実行すれば、画面を横64、縦48に分割した各ブロックごとに点として表示させたり色を塗ったりできる。ただし、LINE命令やCIRCLE命令を使用したときの座標位置(X，Y)は、SCREEN　2と同じく、左上が（0，0)、右下が（255,191）になる。
SCREEN　4	SCREEN　2と同じ256×192ドットの高解像度グラフィックモードだが、COLOR　SPRITE命令が使用でき、スプライトパターンの横1列ごとに色を指定できる。8ドットを1単位とする領域ごとに2色まで使用できる。
SCREEN　5	256×212ドットのビットマップグラフィックモードで、1ドットごとに16色のパレットから自由に選択して色を使用できる。
SCREEN　6	512×212ドットのビットマップグラフィックモードで、1ドットごとに4色のパレットから自由に選択して色を使用できる。
SCREEN　7	512×212ドットのビットマップグラフィックモードで、1ドットごとに16色のパレットから自由に選択して色を使用できる。
SCREEN　8	256×212ドットのビットマップグラフィックモードで、1ドットごとに256色の固定色から自由に選択して色を使用できる。

注

1　SCREEN　0と1は、画面にプログラムリストやメッセージなどを表示させるテキストモードである。グラフィック表示はできない。

2　SCREEN　2と3は、画面にグラフィックを表示するときに使用する。

プログラムやメッセージを表示させることはできない。

3　ＳＣＲＥＥＮ　０のとき、スプライト機能は使用できない。

4　グラフィックモードは、ＩＮＰＵＴ文を実行したり、コマンドレベルに戻ると、自動的にテキストモードとなる。

5　ＳＣＲＥＥＮ　７、８のビットマップモードでは、ＶＲＡＭ128Ｋバイト以上の機種に限り、指定できる。

6　グラフィック画面では、プログラムの実行を終了すると、自動的にテキスト画面に戻るようになっている。今までのグラフィック画面は消えてしまうので、保持したい場合には、ループ行を設ける。

●スプライトサイズの指定

スプライトパターンの大きさを指定する。

スプライトサイズの書式例

ＳＣＲＥＥＮ　，ｎ　　　　ＳＣＲＥＥＮ　１，ｎ

ｎは０～３の数値。左の例はスプライトサイズだけを指定する書式、右の例は画面モードを１に指定してスプライトサイズを指定する書式である。ｎの０～３の意味はつぎのとおり。

ＳＣＲＥＥＮ　,0　　スプライトパターンを８×８ドット標準に指定。
ＳＣＲＥＥＮ　,1　　スプライトパターンを８×８ドット拡大に指定。
ＳＣＲＥＥＮ　,2　　スプライトパターンを16×16ドット標準に指定。
ＳＣＲＥＥＮ　,3　　スプライトパターンを16×16ドット拡大に指定。

スプライトパターンのサイズは、ＳＣＲＥＥＮ　１、ＳＣＲＥＥＮ　２と同時に指定できる。画面モードの指定を省略したとき（上例）はＳＣＲＥＥＮ　１が設定される。

●キークリックの指定

キーボードのキーを押したとき、キークリック音を出せるかどうかを指定する。初期設定はクリック音が出るようになっている。

書式例　ＳＣＲＥＥＮ,,ｎ

ｎは０か１である。０はキークリック音を出さない。１はキークリック音を出す。

●カセットのボーレート指定

カセットにプログラムをセーブするときの転送速度（ボーレート）を指定する。CSAVE文でのボーレート指定が省略された場合、ここで指定したボーレートが有効となる。ただし、カセットをロードするときのボーレートは自動的に判断されるから、改めて指定しなくていい。初期値は1200ボーである。

書式例　SCREEN　,,,n

nは1か2である。1は1200ボー、2は2400ボーである。

注

1　ボーレート2400のときは、テープの質によってはセーブエラーとなる場合がある。

●プリンタのモード指定

プリンタの文字構成が、グラフィック文字やひらがなを持っているMSXの標準プリンタかどうかを指定する。標準プリンタでない場合は、グラフィック文字は空白となり、ひらがなはカタカナに変換される。

書式例　SCREEN ,,,,n

nは0か0以外である。MSX標準プリンタの場合は0、そうでない場合は0以外の数字である。

●ビットマップ

MSXのグラフィック画面は、高解像度グラフィックモードと低解像度グラフィックモードの2種類だけだったが、MSX2では、このほかに、ビットマップグラフィックモードが増設されている。このモードでは、横256×縦212、横

高解像度グラフィック画面

0 1 2 3 4 5 6

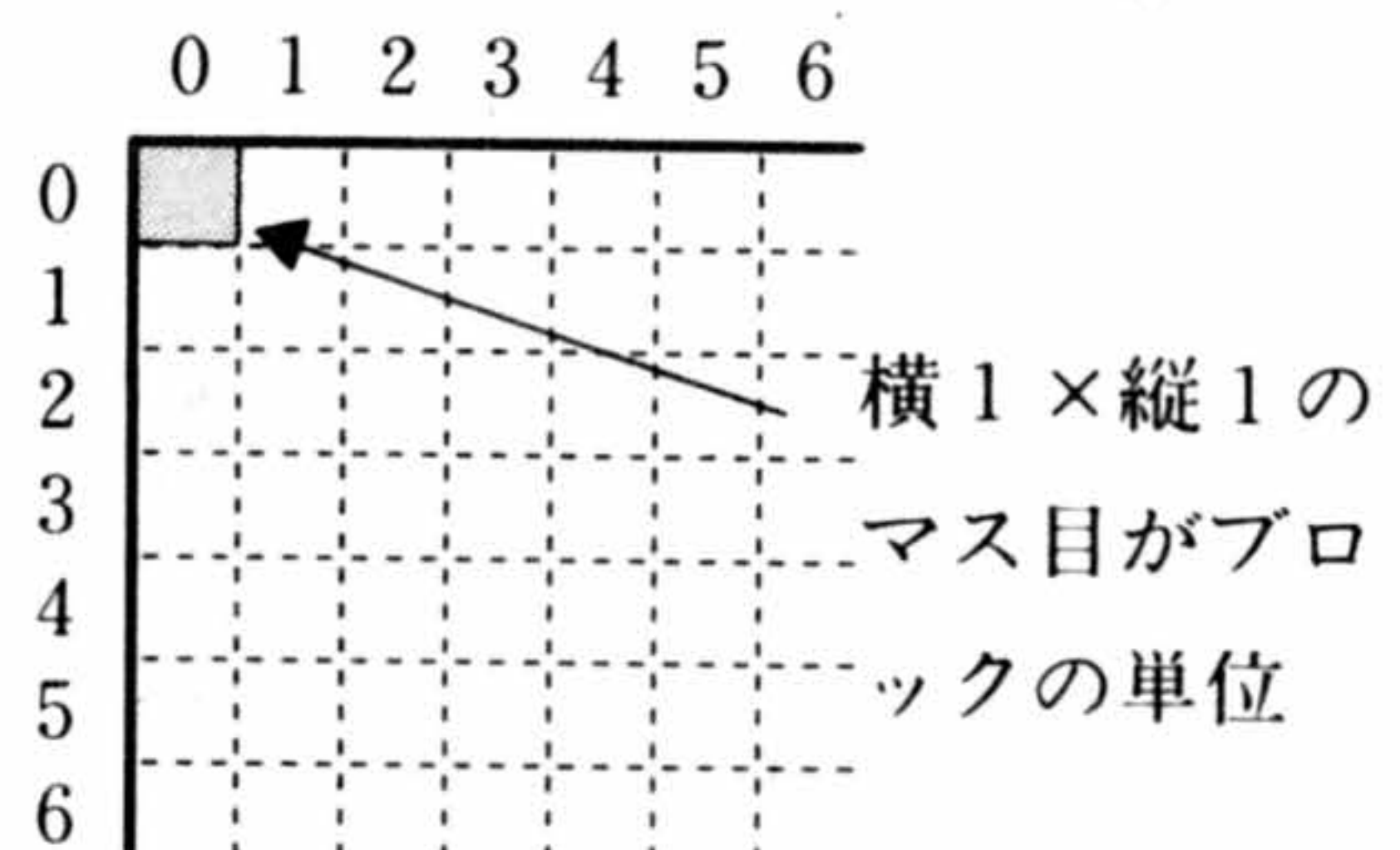
0 1 2 3 4 5 6

1ブロックは横4×縦4ドット

ビットマップグラフィック画面

0 1 2 3 4 5 6

0 1 2 3 4 5 6

横1×縦1のマス目がブロックの単位

(SCREEN　5～8)

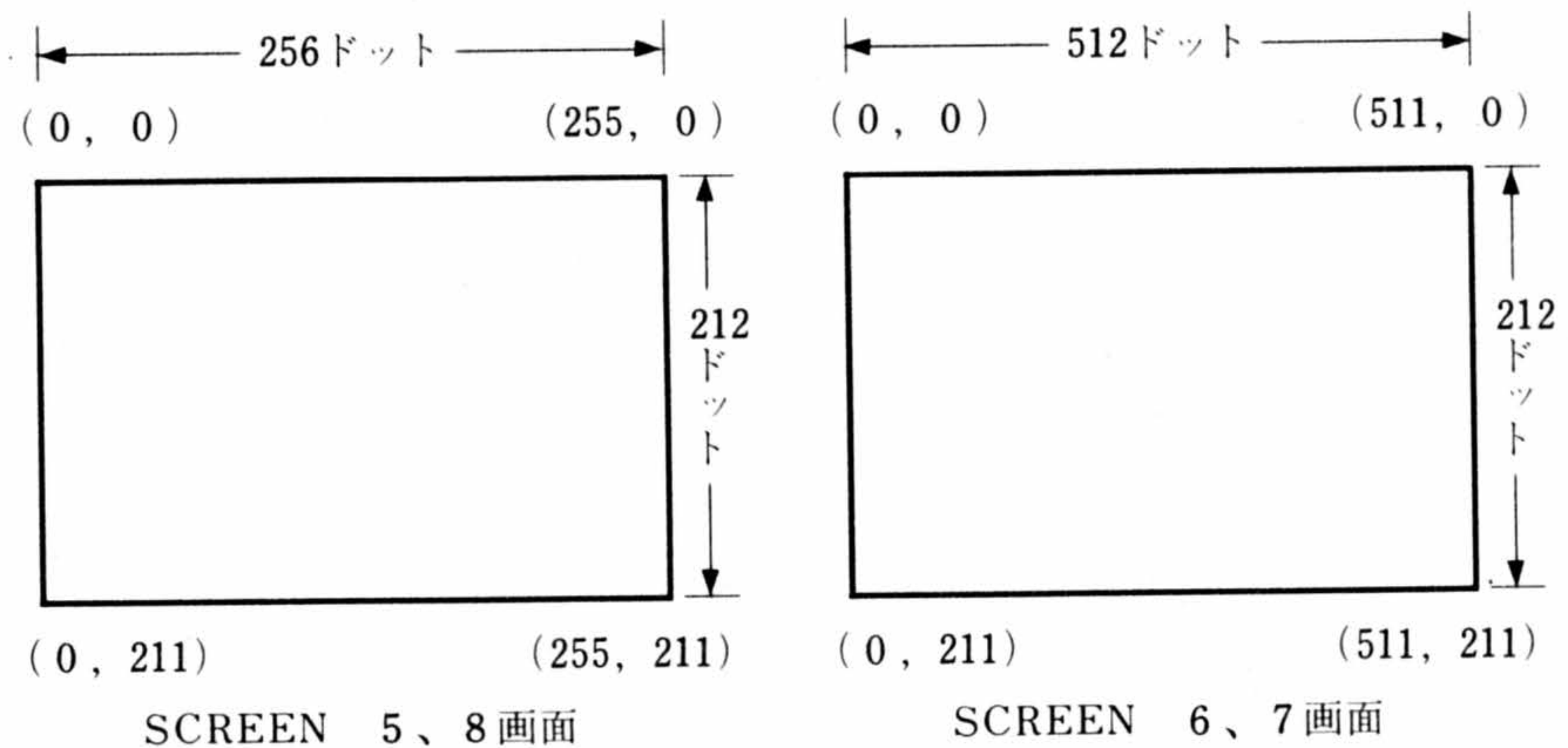
512×縦212の点を使って、より細かな絵や図形を描くことができる。

また、ひとつひとつの点ごとに、512色中16色および256色の色をつけることができ、より鮮明なグラフィック画面を楽しめるようになった。

通常、テキスト画面ではＳＣＲＥＥＮ　1を、グラフィック画面ではＳＣＲＥＥＮ　5~8を使用するとよい。

ただし、ＳＣＲＥＥＮ　7、8は、ＶＲＡＭ128Ｋバイト以上の機種に限るので、注意する。

SCREEN　5、8画面

SCREEN　6、7画面

ビットマップグラフィックモードは、グラフィック画面を同時に何枚も持っており、その中の1枚を選んで表示することができる。

そのときの画面モードとページ数を表にまとめると、以下のようになる。

	VRAM64Kバイトの場合	VRAM128Kバイトの場合
5モード	2ページ	4ページ
6モード	2ページ	4ページ
7モード		2ページ
8モード		2ページ

SET PAGE(セットページ)

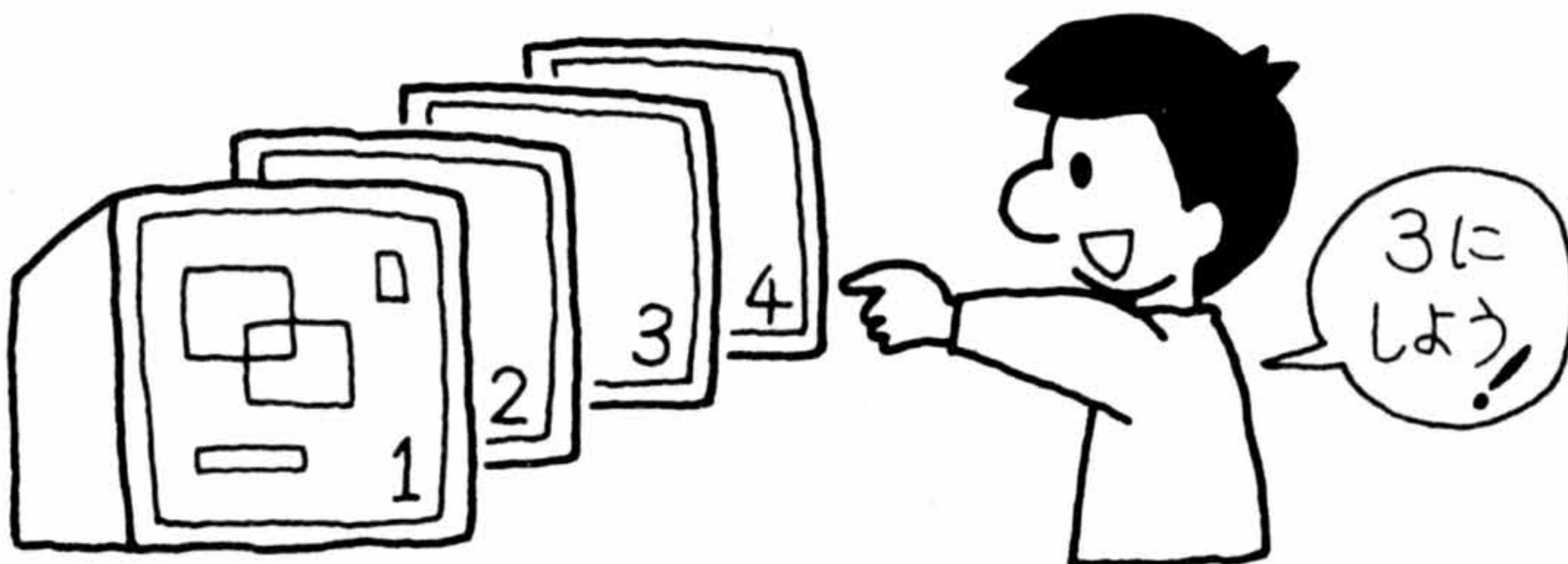
■グラフィック画面のページを切り換える

書式 SET　PAGE [〈ディスプレイページ〉] [,〈アクティブページ〉]

説明 SCREEN5〜8のビットマップグラフィックを、モード内での複数ページに切り換える。〈ディスプレイページ〉は表示されるページで、〈アクティブページ〉は、他のグラフィック命令で描画を行なう対象となるページである。

複数のページを持つ場合、VRAMの容量によってページ数が異なる。

画面モード	VRAM容量	
	64Kバイト	128Kバイト
5	0〜1ページ	0〜3ページ
6	0〜3ページ	0〜3ページ
7	不可	0〜1ページ
8	不可	0〜1ページ

SCREEN命令で画面モードを変更した直後は、SET　PAGE命令でページを指定しないかぎり、ディスプレイページ、アクティブページともに0ページになる。

CHR$(キャラクタダラー)

キャラクタコードから文字を引き出す

キャラクタコードの数値を入力して、そこから文字や記号を得る。

書式 CHR$　(キャラクタコード)

説明 指定するキャラクタコードは、得たい文字や数字につけられたキャラクタコードナンバーである。指定した数値が 0～255 の範囲でない場合Illegal function call のエラーとなる。

書式例 A$=CHR$ (65) ……… キャラクタコード65 (10進コード) に相当する文字 "A" をA$に代入する。

ただし、キャラクタコードの値が 0～31 (10進コード) のコントロールコードとして定義されていると、その機能が実行されて、キャラクタコードに対応する文字は表示されない。

(例)
```
10 PRINT "CHR$(32) カラ CHR$(255) ノ ヒョウジ"
20 PRINT
30 FOR I=32 TO 255
40 PRINT CHR$(I);
50 NEXT: PRINT:PRINT
60 PRINT "CHR$(1) +CHR$(65)カラ"
70 PRINT "CHR$(1) +CHR$(95)マデノ ヒョウジ"
80 PRINT
90 FOR I=65 TO 95
100 PRINT CHR$(1) +CHR$(I);
110 NEXT
```

上のプログラムは、キャラクタコード32～255 までのキャラクタを画面に順次表示させる。ただし、キャラクタコード32と127 、144 、160 、254 、255 はヌル (空白) が表示される。

なお、CHR$はキャラクタコードを指定して、その番号に相当する文字や記号を得るが、これとは逆に、文字を入力してキャラクタコードに変換するの

がＡＳＣ命令である。（ＡＳＣの項を参照）

●グラフィック・キャラクタの表示

キーボードにはグラフィック・キャラクタもある。グラフィック・キャラクタもそれぞれグラフィック・キャラクタコード番号を持っている。このグラフィック・キャラクタは、ＣＨＲ＄を使って画面に表示させることができる。

書式 ＣＨＲ＄（1）＋ＣＨＲ＄(64＋コード番号)

説明 ＣＨＲ＄(1)は、グラフィックキャラクタがあるという符号で、コンピュータにそのことを知らせている。そのあとにＣＨＲ＄(64 ＋コード番号）を加える。ただのグラフィック・キャラクタのコード番号ではなく、64を加えるところが重要である。

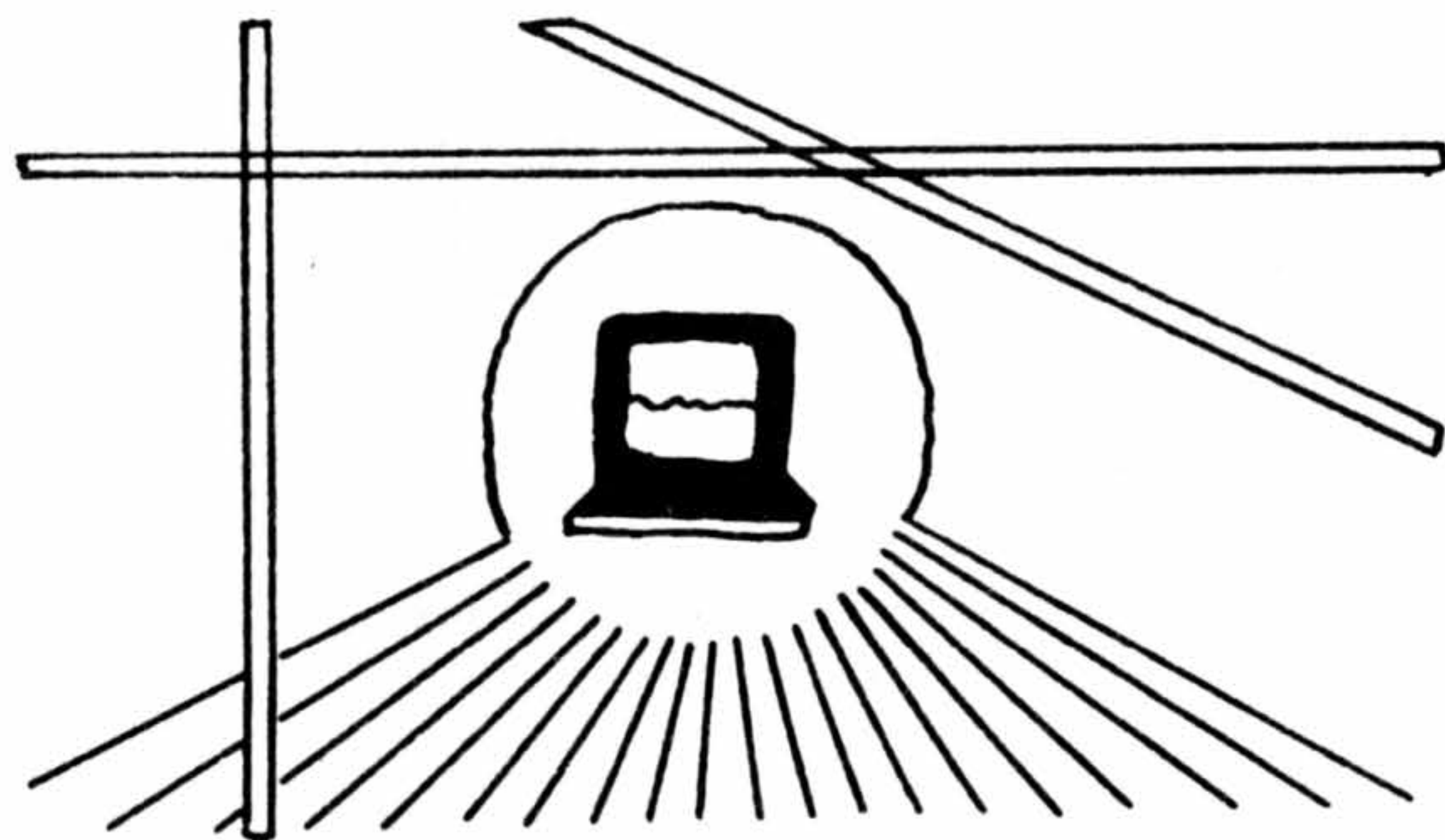
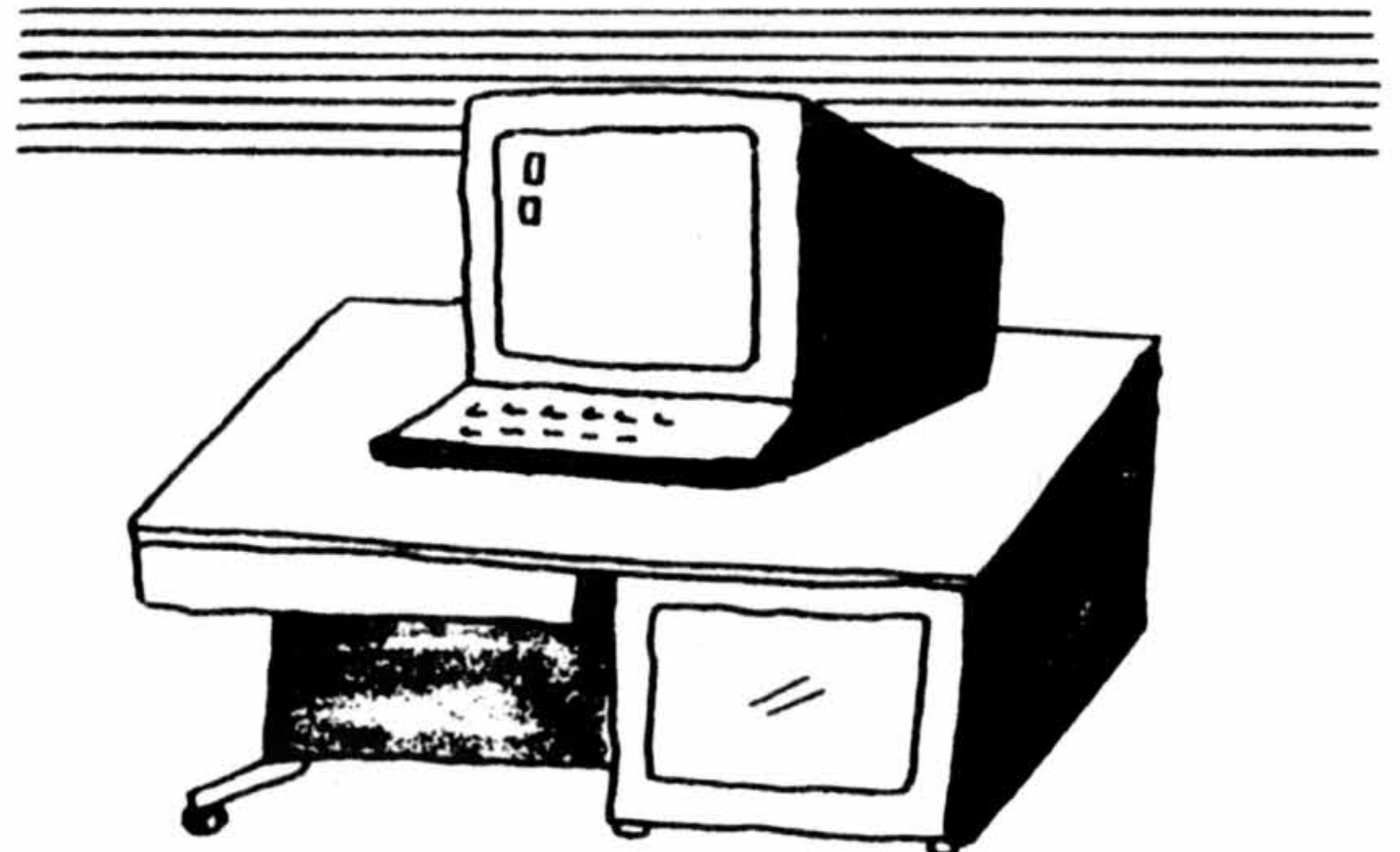
書式例 ＰＲＩＮＴ　ＣＨＲ＄(1)＋ＣＨＲ＄(64+5)

グラフィック・キャラクタ“金”の文字を画面に表示させる。

★★★グラフィック・キャラクタコード表★★★

コード（10進）	コード（16進）	キャラクタ	コード（10進）	コード（16進）	キャラクタ
0	00		16	10	π
1	01	月	17	11	□
2	02	火	18	12	□
3	03	水	19	13	□
4	04	木	20	14	□
5	05	金	21	15	□
6	06	土	22	16	□
7	07	日	23	17	□
8	08	年	24	18	□
9	09	円	25	19	□
10	0A	時	26	1A	□
11	0B	分	27	1B	□
12	0C	秒	28	1C	⊠
13	0D	百	29	1D	大
14	0E	千	30	1E	中
15	0F	万	31	1F	小

★★★キャラクタコード表★★★

コード(10進)	コード(16進)	キャラクタ	コード(10進)	コード(16進)	キャラクタ	コード(10進)	コード(16進)	キャラクタ	コード(10進)	コード(16進)	キャラクタ
0	00	↑	32	20	·	64	40	@	96	60	'
1	01		33	21	!	65	41	A	97	61	a
2	02		34	22	"	66	42	B	98	62	b
3	03		35	23	#	67	43	C	99	63	c
4	04		36	24	$	68	44	D	100	64	d
5	05	コ	37	25	%	69	45	E	101	65	e
6	06		38	26	&	70	46	F	102	66	f
7	07	ン	39	27	'	71	47	G	103	67	g
8	08		40	28	(	72	48	H	104	68	h
9	09	ト	41	29	)	73	49	I	105	69	i
10	0A		42	2A	*	74	4A	J	106	6A	j
11	0B	ロ	43	2B	+	75	4B	K	107	6B	k
12	0C		44	2C	,	76	4C	L	108	6C	l
13	0D	ー	45	2D	−	77	4D	M	109	6D	m
14	0E		46	2E	.	78	4E	N	110	6E	n
15	0F	ル	47	2F	／	79	4F	O	111	6F	o
16	10		48	30	0	80	50	P	112	70	p
17	11	キ	49	31	1	81	51	Q	113	71	q
18	12		50	32	2	82	52	R	114	72	r
19	13	ャ	51	33	3	83	53	S	115	73	s
20	14		52	34	4	84	54	T	116	74	t
21	15	ラ	53	35	5	85	55	U	117	75	u
22	16		54	36	6	86	56	V	118	76	v
23	17	ク	55	37	7	87	57	W	119	77	w
24	18		56	38	8	88	58	X	120	78	x
25	19	タ	57	39	9	89	59	Y	121	79	y
26	1A		58	3A	:	90	5A	Z	122	7A	z
27	1B		59	3B	;	91	5B	[	123	7B	{
28	1C		60	3C	<	92	5C	¥	124	7C	|
29	1D		61	3D	=	93	5D	]	125	7D	}
30	1E		62	3E	>	94	5E	^	126	7E	~
31	1F	↓	63	3F	?	95	5F	_	127	7F	

コード（10進）	コード（16進）	キャラクタ	コード（10進）	コード（16進）	キャラクタ	コード（10進）	コード（16進）	キャラクタ	コード（10進）	コード（16進）	キャラクタ
128	80	♠	160	A0		192	C0	タ	224	E0	た
129	81	♥	161	A1	。	193	C1	チ	225	E1	ち
130	82	♣	162	A2	「	194	C2	ツ	226	E2	つ
131	83	♦	163	A3	」	195	C3	テ	227	E3	て
132	84	○	164	A4	、	196	C4	ト	228	E4	と
133	85	●	165	A5	・	197	C5	ナ	229	E5	な
134	86	を	166	A6	ヲ	198	C6	ニ	230	E6	に
135	87	ぁ	167	A7	ァ	199	C7	ヌ	231	E7	ぬ
136	88	ぃ	168	A8	ィ	200	C8	ネ	232	E8	ね
137	89	ぅ	169	A9	ゥ	201	C9	ノ	233	E9	の
138	8A	ぇ	170	AA	ェ	202	CA	ハ	234	EA	は
139	8B	ぉ	171	AB	ォ	203	CB	ヒ	235	EB	ひ
140	8C	ゃ	172	AC	ャ	204	CC	フ	236	EC	ふ
141	8D	ゅ	173	AD	ュ	205	CD	ヘ	237	ED	へ
142	8E	ょ	174	AE	ョ	206	CE	ホ	238	EE	ほ
143	8F	っ	175	AF	ッ	207	CF	マ	239	EF	ま
144	90		176	B0	ー	208	D0	ミ	240	F0	み
145	91	あ	177	B1	ア	209	D1	ム	241	F1	む
146	92	い	178	B2	イ	210	D2	メ	242	F2	め
147	93	う	179	B3	ウ	211	D3	モ	243	F3	も
148	94	え	180	B4	エ	212	D4	ヤ	244	F4	や
149	95	お	181	B5	オ	213	D5	ユ	245	F5	ゆ
150	96	か	182	B6	カ	214	D6	ヨ	246	F6	よ
151	97	き	183	B7	キ	215	D7	ラ	247	F7	ら
152	98	く	184	B8	ク	216	D8	リ	248	F8	り
153	99	け	185	B9	ケ	217	D9	ル	249	F9	る
154	9A	こ	186	BA	コ	218	DA	レ	250	FA	れ
155	9B	さ	187	BB	サ	219	DB	ロ	251	FB	ろ
156	9C	し	188	BC	シ	220	DC	ワ	252	FC	わ
157	9D	す	189	BD	ス	221	DD	ン	253	FD	ん
158	9E	せ	190	BE	セ	222	DE	゛	254	FE	
159	9F	そ	191	BF	ソ	223	DF	゜	255	FF	

●サンプルプログラム

```
10 CLS
20 COLOR 2,4,1
30 S1$=CHR$(&B00011000)
40 S2$=CHR$(&B00111100)
50 S3$=CHR$(&B00111100)
60 S4$=CHR$(&B01100110)
70 S5$=CHR$(&B11111111)
80 S6$=CHR$(&B01011010)
90 S7$=CHR$(&B11000011)
100 S8$=CHR$(&B00000000)
110 SPRITE$(0)=S1$+S2$+S3$+S4$+S5$+S6$+
S7$+S8$
120 SCREEN 2
130 FOR M=1TO30
140 X=RND(1)*256:Y=RND(1)*192:C=RND(1)*
15+1
150 PSET(X,Y),3
160 NEXT
170 SOUND7,54:SOUND6,14:SOUND8,15
180 FOR N=190TO-0STEP -1
190 PUT SPRITE 0,(100,N),9,0
200 FOR M=1TO 50:NEXT
210 NEXT
220 PLAY"A"
```

③一般ステートメント

GOTO（ゴーツー）

■指定した行番号へジャンプ

プログラムの順序を変えて、行番号で指定する行にジャンプさせ、その行以降の命令を実行させる。

書式 GOTO 行番号

（GO TO 行番号）

説明 指定した行番号へ無条件で進めという意味で、無条件ジャンプともいわれている。

指定する行番号は、現在実行しているプログラムにつけてある行番号でなければならない。もし、指定した行番号がプログラムの中になければ "Undefined line number" と画面表示され、その後に行番号が表示される。

また、"GOTO" は "GO TO" と書いてもいい。ただし、GOとTOの間のスペースは1個だけ。2個以上スペースをあければ、GOTO命令と判断されない。

書式例 GOTO 20 ………… 指定行番号20へジャンプする。

（例）
```
10 CLS
20 A=0
30 A=A+1
40 PRINT A
50 GOTO 30
```

このプログラムを実行させると、画面には数字が1から順に表示され続ける。つまり、50行までくるとGOTO 30で30行へ行き30〜50行の間のプログラムの実行を繰り返す。

（例）
```
10 PRINT "ABC" :GOTO 10
```

このプログラムは、画面に“ABC”と表示させたあと、行番号10へ行けという意味で、このプログラムを実行すると限りなく繰り返す。こうしたプログラムを、無限ループという。

このプログラムを止めるときは、CTRL+STOPキーを同時に押せば、無限ループから抜け出して止まる。

注

1　GOTO文のあとには行番号を示す数字以外、変数や式を書くことはできない。変数を書くと、Syntax errorとなり、プログラムが止まる。

2　GOTO文は、プログラム中に多く使用される。しかし、あまり多用しすぎると、プログラムの流れがあっちこっちに行って複雑になってしまう。

GOTO文の多いプログラムを一般にはスパゲティプログラムといい、すぐれたプログラムとはいえない。GOTO文はあまり多用せずプログラムの簡素化を計るべきである。簡素化されたプログラムは、虫取り（デバッグ）をするときにも、間違いが発見しやすいし、また他人が見ても理解しやすい。

●サンプルプログラム

```
10 CLS
20 COLOR 10,1,8
30 KEY OFF
40 GOTO170
50 A=INT(RND(-TIME)*9+1)
60 LOCATE10,10:PRINT"*****"
70 LOCATE10,11:PRINT"*   *"
80 LOCATE10,12:PRINT"* ? *"
90 LOCATE10,13:PRINT"*   *"
100 LOCATE10,14:PRINT"*****"
110 LOCATE10,16:INPUTB
120 IF A=B THEN GOTO140 ELSE150
130 STOP
140 PRINT"ko":FOR N=1TO500:NEXT:CLS:GOTO50
150 PRINT"No":LOCATE11,12:PRINTA
160 FOR N=1TO500:NEXT:CLS:GOTO50
170 CLS
180 PRINT"     ｹﾞｰﾑ ﾉ ｾﾂﾒｲ"
190 PRINT:PRINT
200 PRINT"  1-10 ﾏﾃﾞﾉ ｽｳｼﾞ ｦ ｱﾃﾃ ｸﾀﾞｻｲ"
210 PRINT:PRINT"OK {Y} ｦ ｵｽ"
220 A$=INKEY$
230 IF A$="y"THENCLS:GOTO50
240 IF A$="Y"THENCLS:GOTO50
250 GOTO 220
```

GOSUB～RETURN(ゴーサブ～リターン)

■指定した行番号のサブルーチンを呼び出す

プログラムの中で何度も繰り返し使用するブロックをサブルーチンとして独立させ、必要に応じて呼び出して使用すれば便利である。GOSUB～RETURN命令はGOSUB命令でそのサブルーチンを呼び出し、RETURN命令でもとに戻る。

書式 GOSUB 行番号
～
RETURN 行番号

説明 GOSUBで指定した行番号から始まるサブルーチンにジャンプして処理したあと、RETURN命令でGOSUB文の次のプログラム文に戻る。もちろん、GOSUBで指定する行番号は、変数や式は使用できない。

RETURN命令の後の行番号は、指定しない場合が多く、GOSUB文の次の文に戻る。RETURN文で行番号を指定すれば、強制的にその行番号に戻る。

書式例 GOSUB 100 ……… 行番号100から始まるサブルーチンへジャンプする。
～
RETURN 300 ……… 300と指定しなければGOSUB文の次の文へ戻る。この例では300と指定されているから、300行へ戻る。

(例)
```
10 A=10:B=20
20 GOSUB 60
30 A=30:B=16
40 GOSUB 60
50 END
60 C=A+B        ┐
70 PRINT C      ├ サブルーチン
80 RETURN       ┘
```

```
RUN
 30
 46
 OK
■
```

左ページのプログラムは20行と40行でＧＯＳＵＢ文を使用し、ともに60行から始まるサブルーチンをコールしている。このプログラムで60〜80行がサブルーチンである。このように何回も使用するプログラムのブロックはサブルーチンにすればいい。サブルーチンにせず同じプログラムを何回も入力するのは、めんどうなばかりかメモリのむだ使いになる。

プログラム80行のＲＥＴＵＲＮ命令では、20行からのコールのとき30行に戻り、40行からのコールのときは50行に戻る。

注

1　ＧＯＳＵＢで呼び出したサブルーチンからさらにＧＯＳＵＢで呼び出すこともできる。これをプログラムの多重化というが、あまり多重化をすすめるとプログラムが複雑になりすぎる。多重化はメモリのスタック領域のメモリ容量が許すかぎりできるが、容量を越えると、"out of memory" とエラーメッセージが表示され、プログラムは中断する。

●サンプルプログラム

```
100 CLS
110 FOR N=1 TO 28
120 LOCATEN,10:PRINT"#"
130 GOSUB 200
140 NEXT
150 FOR N=28TO 1 STEP -1
160 LOCATEN,10:PRINT" "
170 GOSUB 230
180 NEXT
190 END
200 FOR I=1TO500
210 NEXT
220 RETURN
230 FOR I=1TO100
240 NEXT
250 RETURN
```

ON～GOTO（オン～ゴーツー）
ON～GOSUB（オン～ゴーサブ）

■指定されたいずれかの行番号に飛ぶ

ONのうしろにくる変数や式の値に応じて、指定した行番号のいずれかにジャンプする。

書式

ON　変数または式　GOTO　行番号1，行番号2，行番号3……………

ON　変数または式　GOSUB　行番号1，行番号2，行番号3……………

説明　ON～GOTOはONのあとにくる変数または式の値に応じて、指定された行番号のなかのどこに飛ぶかが指定されている。もし、変数値が1なら行番号1へ、変数値が2なら行番号2へ、3なら行番号3へ……とジャンプする。

指定する行番号の数は何個でもいいが、必ずコンマ（,）で区切る。

ON～GOSUB文で指定する行番号は、ジャンプするサブルーチンの最初の行である。サブルーチンを実行したあと、RETURNでON～GOSUBのつぎの行に戻ってくる。

書式例　ON　A　GOTO　100,200,300……　変数Aの値が1のときは100行へ、2のときは200行へジャンプ。

ON　A　GOSUB　100,200,300……　変数Aの値が2のときは200行へ、3のときは300 行へそれぞれジャンプ。

変数Aの値が0の場合は、ON～GOTO、ON～GOUSB文のつぎの行番号を実行する。

また、変数値が指定した行番号の個数より多い場合(ここでは4以上)にも、0と同様につぎの行番号を実行する。

注

1　ONのあとの変数や式の値が整数でない場合は、小数点が切り捨てられた

値で実行される。

2　ONのあとの変数や式の値が負の数になった場合は、“Illegal function call”というエラーメッセージが画面表示され、プログラムの実行はストップする。

●サンプルプログラム

```
100 CLS
110 COLOR 10,1,8
120 KEY OFF
130 PRINT"1-3 ノ スウジ゛ ヲ オス"
140 A$=INKEY$
150 A=VAL(A$)
160 ON A GOSUB 180,200,220
170 GOTO 140
180 PRINT"1 ハ msx "
190 RETURN
200 PRINT"2 ハ abc "
210 RETURN
220 PRINT"3 ハ xyz "
230 RETURN
```

INPUT（インプット）

■キー入力のデータを変数に代入する

キーボードからデータを入力するときに使用する。そのデータはあとで指定した変数に代入される。

書式　INPUT　"注釈文"；変数名，変数名……

説明　INPUT文を実行すると、画面には？（クエッションマーク）と1桁のスペースが表示されて、キーボードからの入力待ちの状態になる。このあとキーボードからデータを入力してRETURNキーを押せば、そのデータが変数に代入される。注釈文（プロンプト文）を書いておけば、？の前に注釈文が表示される。

変数はコンマ（,）で区切って、いくつでも入力できる。指定した変数の個数よりキー入力データが少ない場合は、？？と表示され、キー入力待ちの状態になる。

また、入力したデータが対応する変数のデータの型と一致しない場合は、画面に"?Redo from start"と表示され、再入力状態になる。

書式例　ＩＮＰＵＴ　"ナマエ　ハ"；Ａ＄……　名まえをキー入力してＡ＄に代入する例。

（例）

```
10 CLS
20 INPUT "ナマエ ハ";A$
30 PRINT A$
40 END
```

```
ナマエ ハ? MSX
MSX
OK
■
```

注

1　グラフィックモードでＩＮＰＵＴ文を使用すると、自動的にテキストモードになる。

ＩＮＫＥＹ＄(インキーダラー)

■１文字のキー入力データを指定した変数に代入

キーボードから入力した文字や数字のデータの１文字だけを取り出して、それを関数の値とする。

書式　変数＝ＩＮＫＥＹ＄

説明　１文字のキー入力データを変数に代入するが、プログラムはこの命令箇所でストップせず実行を続ける。キー入力がない場合には変数にヌルストリング（空白）を返して通過してしまう。

書式例　Ａ＄＝ＩＮＫＥＹ＄

（例）10　ＣＬＳ　　20　Ａ＄＝ＩＮＫＥＹ＄
　　　30　ＰＲＩＮＴ　Ａ＄　　40　ＧＯＴＯ　20

このプログラムを実行すると、画面では変化しないが、キー入力をすると左側に１文字表示される。

注

1　プログラムにＩＮＫＥＹ＄でＣＴＲＬ＋ＣやＣＴＲＬ＋ＳＴＯＰを拾うことはできない。

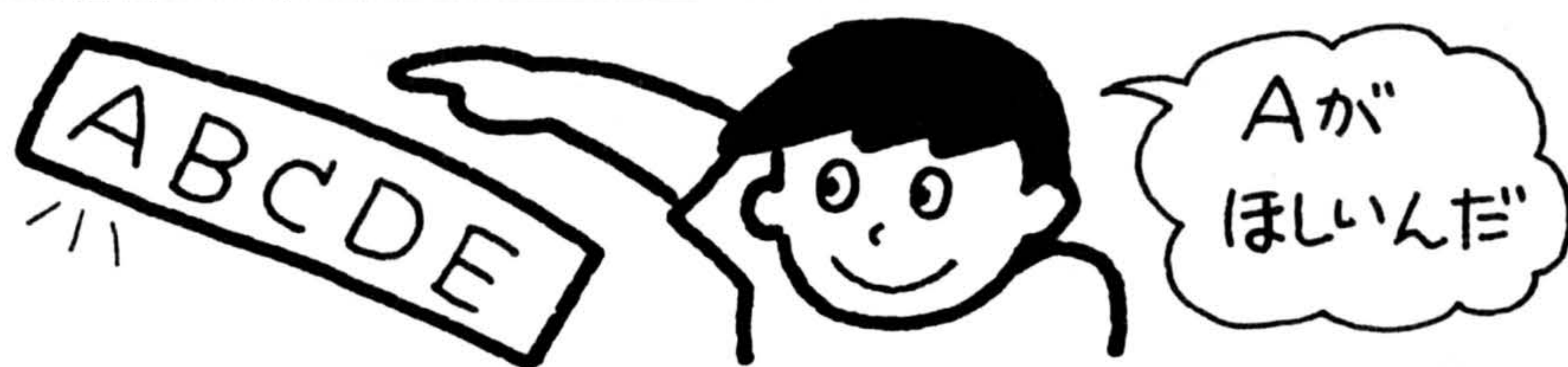
2　プログラムにＩＮＫＥＹ＄があっても実行時は停止しない。

●サンプルプログラム

```
10 CLS
20 PRINT"key ヲ オス"
30 A$=INKEY$
40 FORN=1TO100:NEXT
50 PRINTA$;
60 GOTO30
```

```
10 CLS
20 PRINT"key ヲ オス"
30 A$=INKEY$
40 FORN=1TO100:NEXT
50 PRINTA$;
60 GOTO30
```

ABCDE

Aが
ほしいんだ

ＩＮＰＵＴ＄（インプットダラー）

■キーボードあるいはファイルから文字列を入力

指定されたファイルから指定された長さの文字列を入力する関数である。ファイルバッファ番号が省略されると、キーボードから入力が行なわれる。

書式

①　ＩＮＰＵＴ＄（〈文字数〉） ②　ＩＮＰＵＴ＄（〈文字数〉，［＃］〈ファイルバッファ番号〉）

説明　書式①では、キーボードから入力が行なわれる。ＩＮＰＵＴ文、ＩＮＫＥＹ＄文と似た働きをするが、ＩＮＰＵＴ文と異なり入力された文字は画面に表示されない。また、プログラム実行中、ＩＮＫＥＹ＄文はストップせず実行を続けるのに対して、ＩＮＰＵＴ＄文では、カーソルが画面に表示され、文字数を満たすと実行を再開する。

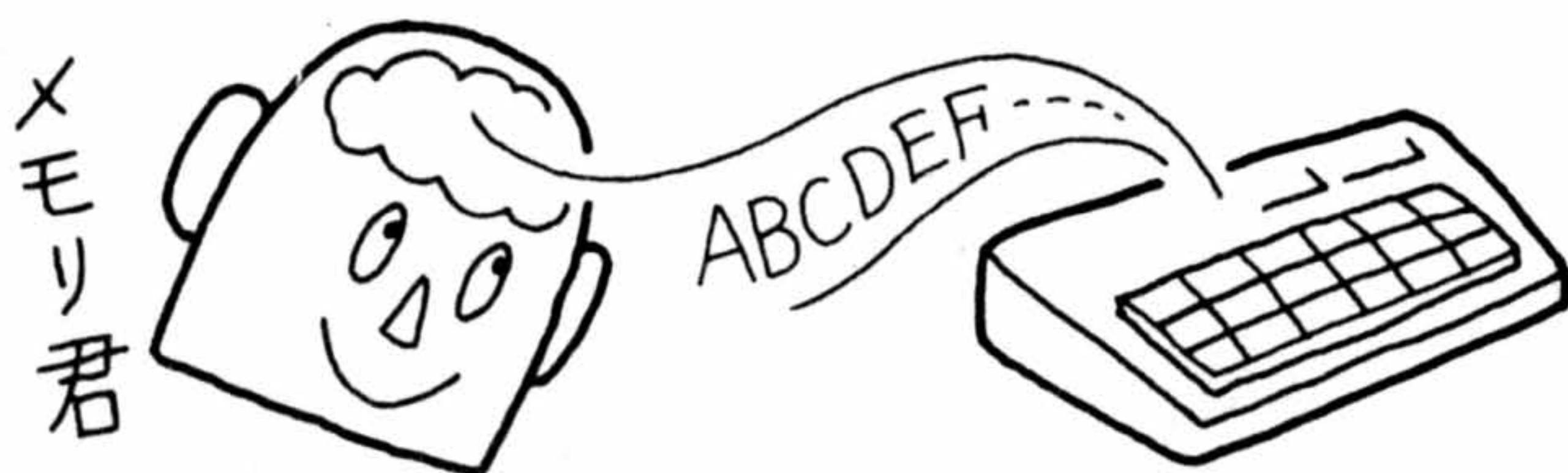
書式②は、ファイルバッファ番号で指定されたファイルから、文字数で指定された長さだけの文字列を読み込んで、その値を返す。

指定するファイルバッファ番号は、ＯＰＥＮ文でファイルを開いたときに指定したファイル番号で、入力モードで開かれていなければならない。

指定できる文字数は、1～255の範囲である。

この関数は、ＬＩＮＥ　ＩＮＰＵＴ＃文でも読み込めない改行コードなどを、ファイルから読み込むときに使うことができる。

書式①②とも、文字数で指定された文字が入力できるまで、カーソルが表示されて入力待ちの状態となり、文字の入力が満たされると、リターンキーを押さなくても自動的に入力が終了する。

書式例　Ａ＄＝ＩＮＰＵＴ＄（１）

IF～THEN～ELSE(イフ～ゼン～エルス)

■条件を判断して実行したりしなかったり

条件が成立しているかどうかを調べ、その条件しだいで実行するプログラムを指定する。

書式 IF　条件　THEN　命令文

　　 IF　条件　THEN　命令文1　ELSE　命令文2

説明 上のIF～THEN文はIFのあとの条件が合っていれば、THEN以降の命令文を実行し、合っていない場合はすぐつぎの行番号を実行する。

IF～THEN～ELSE文はIFのあとの条件が合っていれば、THEN以降の命令文1を実行し、合っていない場合はELSE以降の命令文2を実行する。

ともにTHEN以降のプログラムは、マルチステートメントを使用して、1行が255文字を超えないかぎり何個でも命令を書くことができる。

一般に30行へジャンプさせたいときの命令はGOTO　30と書くが、THENのあとにかぎりGOTOは省略してもいい。

書式例 IF　A=B　THEN　30　……　AとBの値が等しければ30行へジャンプせよ。

IF　A=B　THEN　30　ELSE10

……　AとBの値が等しければ30行へジャンプし、そうでなければ10行へジャンプせよ。

IF以下の条件は、比較演算子を使って2つの数字や変数の値を比較する。

(例)
```
10 INPUT A
20 B=INT (A/2)
30 IF B=A/2 THEN 50 ELSE 60
40 END
50 PRINT A:GOTO 10
60 PRINT "2デワレマセン":GOTO 10
```

プログラムを実行すると、画面に“？”マークが出てキーの入力待ちになるから数字をキー入力する。30行で２で割れるかを判断し、Ａの値が２で割れれば50行へ飛び、そうでないと60行に飛んで各処理を行ったあと10行へ行く。

●比較演算子

演算子	意　味	例
①＝②	①と②は等しい	Ａ＝Ｂ
①<>②	①と②は等しくない	Ａ<>Ｂ
①＜②	①は②より小さい	Ａ＜Ｂ
①＞②	①は②より大きい	Ａ＞Ｂ
①>=②	①は②より大きいか等しい	Ａ>=Ｂ
①<=②	①は②より小さいか等しい	Ａ<=Ｂ

（例）ＩＦ Ａ＝Ｂ ＴＨＥＮ ＩＦ Ｂ＝Ｃ ＴＨＥＮ Ｂ＝８ ＥＬＳＥ Ｃ＝６

上の例は、ＡとＢが等しい場合にＢ＝ＣならばＢに８を代入する。Ｂ＝ＣでなければＣに６が代入される。ここで注意する点は、Ａ＝Ｂでない場合はそれ以降は無視されて、つぎの行番号を実行するということである。

ＩＦ文中ではＴＨＥＮの代わりにＧＯＴＯを使用することができる。前に、たとえばＧＯＴＯ　５０の代わりにＴＨＥＮ　５０としてもいいと説明したが、その反対にＴＨＥＮの代わりにＧＯＴＯとしてもいい。

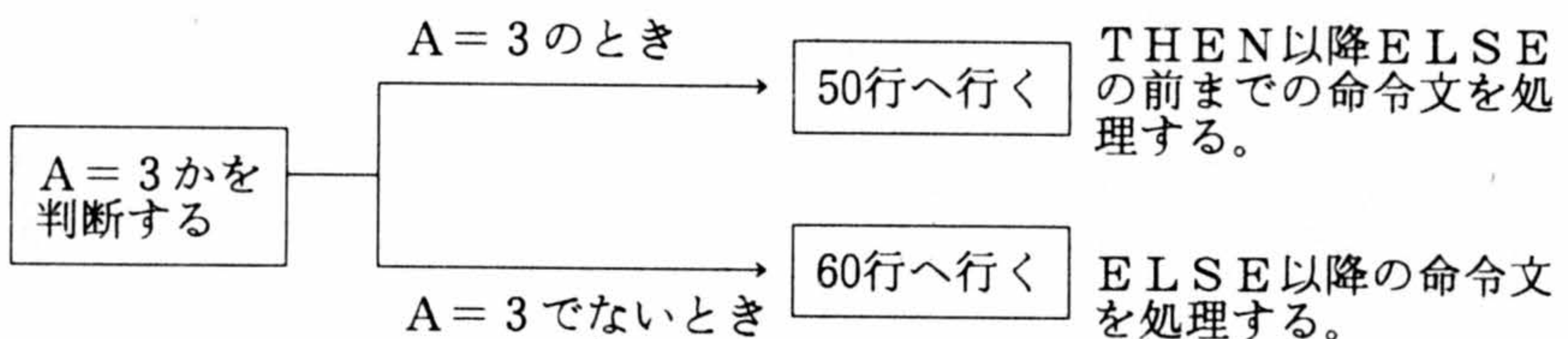
（例）ＩＦ　Ａ＝３　ＴＨＥＮ　５０　ＥＬＳＥ　ＧＯＴＯ　６０
　　　ＩＦ　Ａ＝３　ＧＯＴＯ　５０　ＥＬＳＥ　６０

上の例のように、ＴＨＥＮ　５０をＧＯＴＯ　５０としてもいいし、またＥＬＳＥのあとのＧＯＴＯを省略していい。

上の２つの例のうち、上例よりも下例の方が、ＢＡＳＩＣでの処理が少し早い上、プログラムも短くすることができる。

Ａ＝３かを判断する

Ａ＝３のとき → 50行へ行く　ＴＨＥＮ以降ＥＬＳＥの前までの命令文を処理する。

Ａ＝３でないとき → 60行へ行く　ＥＬＳＥ以降の命令文を処理する。

●いろいろな判断方法

1　文字列の大小比較判断

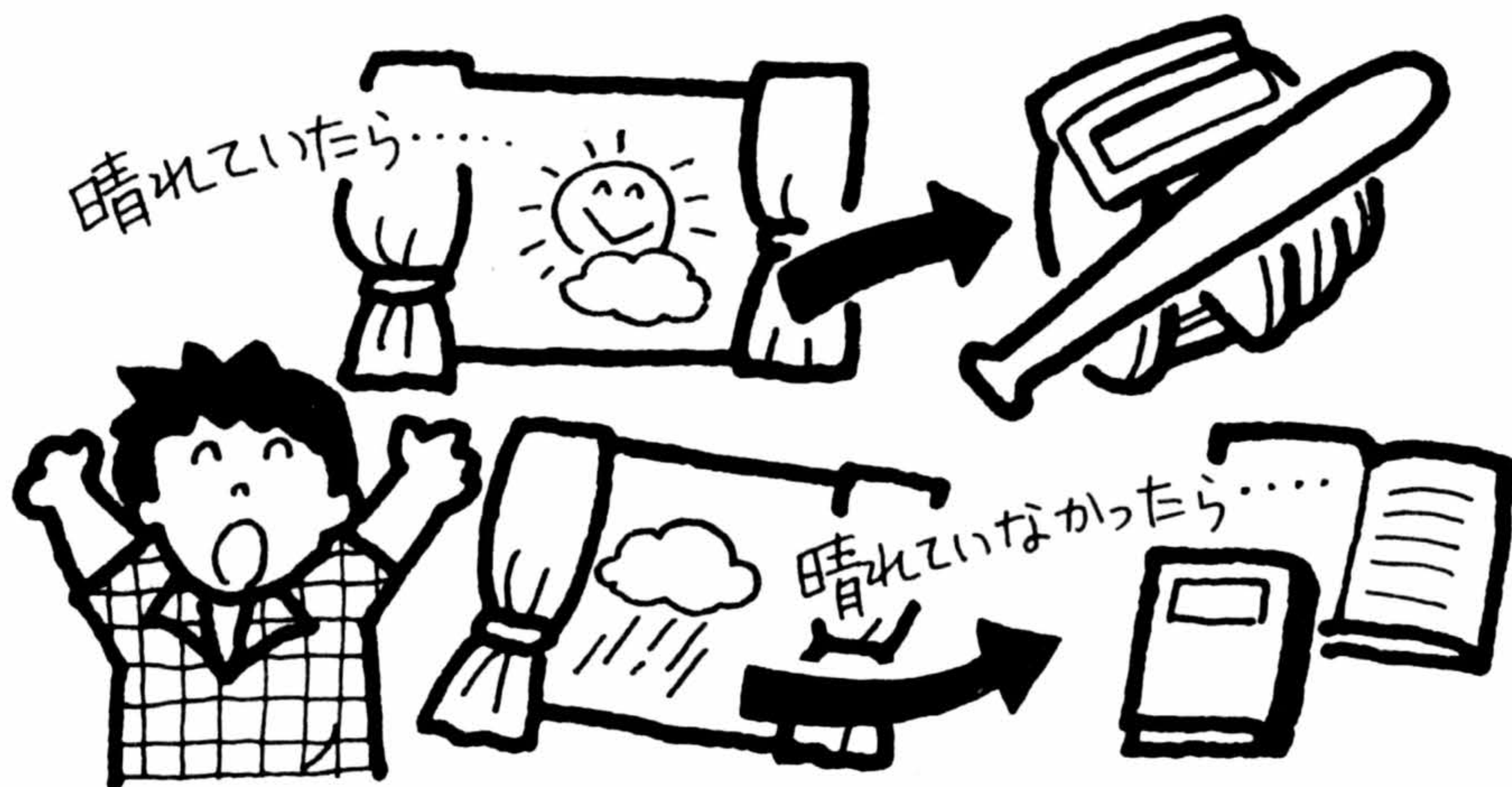
文字列を使って大小の比較をする方法もある。文字列の左側から順に文字を比較していく。全部が同じであれば両方の文字列は等しい。途中で異なった文字があった場合、キャラクタコードの大きい方の文字列が大きい。文字列の長さが異なるときは、長い方が大きい。文字列中の空白もコードとみなされる。

（例）　"ＡＢＣ"　＝　"ＡＢＣ"
　　　　"ＡＢＣ"　<　"ＡＢＤ"
　　　　"ＭＳＸ＆"　>　"ＭＳＸ＃"
　　　　"ｍｓｘ"　>　"ＭＳＸ"
　　　　"ＭＳ　"　>　"ＭＳ"
　　　　"ＭＳＸ"　>　"ＭＸ"

(2)　変数の値が０かどうかの判断

変数の値や数式の値が０であるかどうかを判断する。このときの値が０以外を真とし、０を偽と判断する。

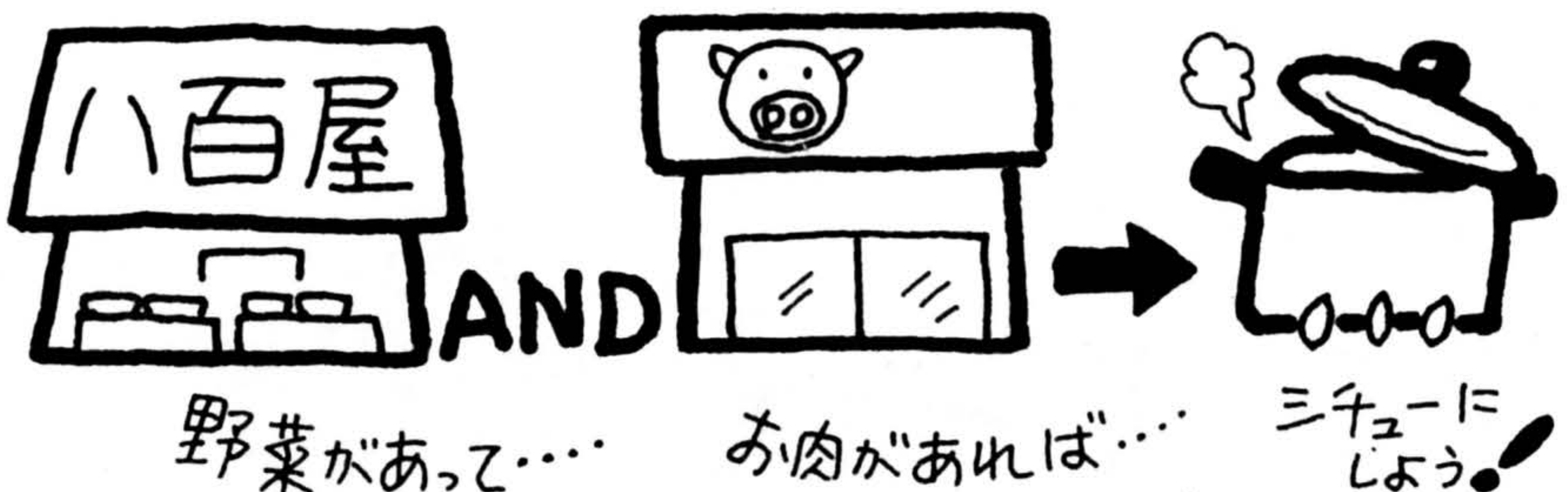
（例）　ＩＦ　ＡＢ　ＴＨＥＮ　200　ＥＬＳＥ　10

もしＡＢという変数の値が 0なら10行へ行き、 0以外なら200 行へ行く。もし、ＥＬＳＥがない場合はつぎの行番号を実行する。

(3)　複合条件判断

複数個の条件を論理的に各演算子を使って判断する。演算子には、ＮＯＴとＡＮＤ、ＯＲの３種類を使用する。

（例）　ＮＯＴ：　ＩＦ　ＮＯＴ（ＡＢ＝０）　ＴＨＥＮ　100
変数ＡＢの値が０でなければ 100行へ行く。
ＡＮＤ：　ＩＦ　Ｘ＝０　ＡＮＤ　Ｙ＝０　ＴＨＥＮ　100
変数ＸとＹの値がともに０なら100 行へ行く。
ＯＲ　：　ＩＦ　Ｘ＝０　ＯＲ　Ｙ＝０　ＴＨＥＮ　100
変数ＸとＹの値のどちらかが０であり、あるいは双方０である場合は100 行へ行く。

●サンプルプログラム

```
100 CLS
110 PRINT"G ﾋﾀﾞﾘ:J ﾐｷﾞ:Y ｳｴ:N ｼﾀ"
120 A$=INKEY$:IF A$=""THEN120
130 IF A$="Y"THENYY=YY-1:GOTO220
140 IF A$="y"THENYY=YY-1:GOTO220
150 IF A$="N"THENYY=YY+1:GOTO220
160 IF A$="n"THENYY=YY+1:GOTO220
170 IF A$="G"THENXX=XX-1:GOTO220
180 IF A$="g"THENXX=XX-1:GOTO220
190 IF A$="J"THENXX=XX+1:GOTO220
200 IF A$="j"THENXX=XX+1:GOTO220
210 GOTO120
220 LOCATEXX,YY:PRINT"$"
230 GOTO120
```

FOR～NEXT（フォア～ネクスト）

■一連の命令を指定回数だけ繰り返す

ＦＯＲ文からＮＥＸＴ文の間をＴＯ後の指定回数だけ繰り返す。

書式 ＦＯＲ　変数＝初期数値・変数　ＴＯ　最終数値・変数

～

ＮＥＸＴ　変数

説明 変数の値が初期数値から１ずつ増えていって、最終数値になるまでＦＯＲとＮＥＸＴの間のプログラムを繰り返す。この場合、初期数値は最終数値よりも小さくなければならない。もし、大きい場合はループとならず抜け出てしまう。

初期数値および最終数値には変数や式が使用可能である。

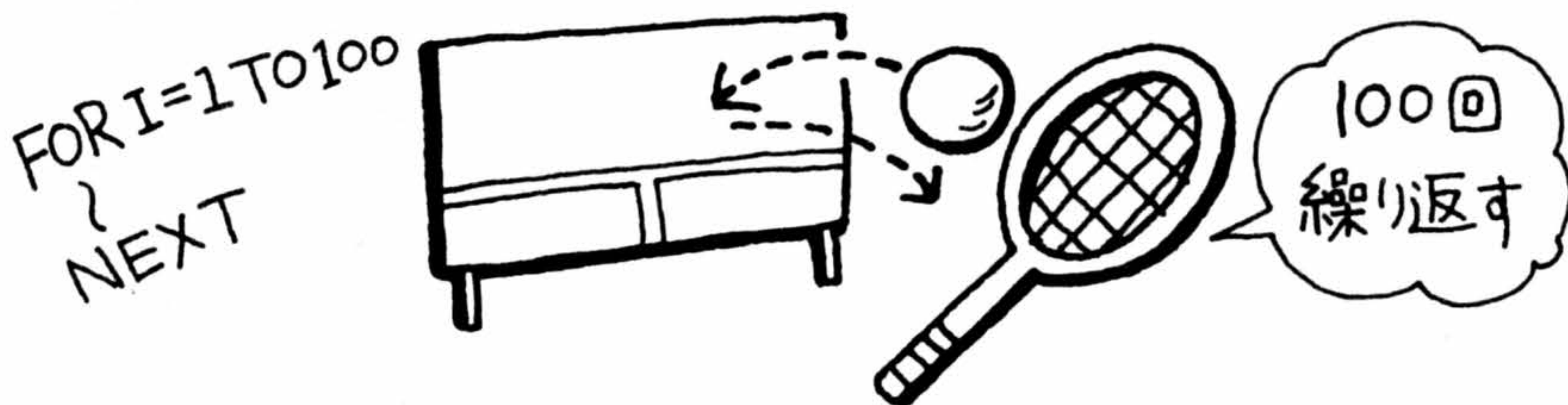
（例）

```
10 FOR N=1 TO 20  ┐
20 PRINT N;       │ループ
30 NEXT N         ┘
40 END
```

上のプログラムを実行すると画面に数字の１～20までが表示される。このように、初期数値１から最終数値２０までを繰り返し、その間に変数Ｎの値を１ずつ増していき、20行で変数Ｎの値を画面に表示する。

30行のＮＥＸＴのあと書かれている変数Ｎは、ＦＯＲのあとの変数と同じでなければならない。違う変数だと画面に “Next without for” とエラーメッセージが出てストップする。また、ＮＥＸＴのあとの変数は省略しても、プログラム自体に問題はないが、ループが２重、３重になった場合、デバッグ処理に手間取ってしまう。

FOR I=1 TO 100
～
NEXT

100回
繰り返す

●２重３重のループについて

ＦＯＲ～ＮＥＸＴ文はプログラムの中で何重にでも使用できる。すなわち、ＦＯＲ～ＮＥＸＴ文の中にＦＯＲ～ＮＥＸＴを何回も使用できる。ただし、ＦＯＲ～ＮＥＸＴのループを交差させてはならない。

（例）　ループが交差していない

```
10 FOR I=0 TO 5 ─────┐
20 FOR J=0 TO 2 ─┐   │
     ～          │   │
50 NEXT J ───────┘   │
60 NEXT I ───────────┘
```

（ループ２、ループ１）

50行、60行はまとめて、50　ＮＥＸＴ　Ｊ，Ｉとしてもいい

（例）　ループが交差している

```
10 FOR I=0 TO 5 ───┐
20 FOR J=0 TO 2 ─┐ │
30 NEXT I ───────┼─┘
                 │
50 NEXT J ───────┘
```

（ループ１、ループ２）

“NEXT without for”のエラーメッセージが表示される

●増分の指定

ＦＯＲ～ＮＥＸＴ文の増分は１ずつだが、ＳＴＥＰを指定すれば、増分を自由に増減できる。

書式　ＦＯＲ　変数＝初期数値　ＴＯ　最終数値　ＳＴＥＰ　増分

説明　ＦＯＲ～ＮＥＸＴの意味は同じだが、ＳＴＥＰで指定した数値ずつ増加し、最終数値でループを抜け出す。

（例）

```
10 FOR I=1 TO 10 STEP 2
20 PRINT I;
30 NEXT I
40 END
```

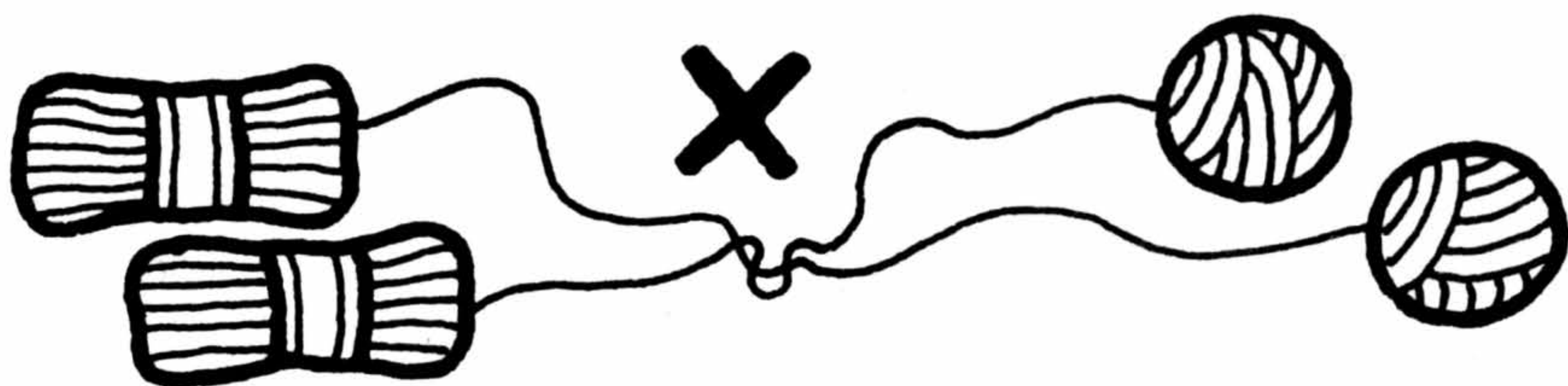
このプログラムを実行すると、"１３５７９"の数字が表示される。変数Ｉが１から始まり、その後２ずつ増加したからである。９に２をたすと11となり最終数値10より大きくなるため、９でループが終わる。

注

1　ＳＴＥＰは増減ともにできる。このときの初期数値は最終数値よりも大きく、ＳＴＥＰの数値には負の記号（－）がつく。

2　初期数値、最終数値やＳＴＥＰ数値には、変数や式も使用できる。

●サンプルプログラム

```
10 CLS
20 COLOR 15,1,8
30 LOCATE 5,0
40 PRINT"ｶﾞｸｼｭｳ ﾌﾟﾛｸﾞﾗﾑ"
50 S=0
60 FOR N=0 TO 18 STEP 2
70 A=INT(RND(-TIME)*9)+1
80 B=INT(RND(-TIME)*9)+1
90 LOCATE 5,2+N
100 PRINTA;"+";B;"="
110 LOCATE14,2+N
120 INPUT Z
130 IFA+B=ZTHENLOCATE25,2+N:PRINTCHR$(132):S=S+1
140 IFA+B<>ZTHENLOCATE25,2+N:PRINTCHR$(133)
150 NEXT
160 PLAY"T250L64CDEFGABO5CCO4BAGFEDC"
170 LOCATE2,21
180 PRINT"10 ﾀﾞｲ ﾁｭｳ";S;"ﾀﾞｲ ｱﾀﾘ ﾏｼﾀ"
190 FOR M=1TO2000:NEXT
200 CLS
210 LOCATE5,10
220 PRINT"ﾓｳｲﾁﾄﾞ ｼﾏｽｶ (Y/N)"
230 A$=INKEY$:IF A$="y"THENRUN
240 IF A$="Y"THENRUN
250 IF A$="n"THEN280
260 IF A$="N"THEN280
270 GOTO 230
280 END
```

STOP（ストップ）

■プログラムの実行を停止させる

プログラムの実行を強制的に停止させ、コマンドレベルに戻す。

書式 STOP

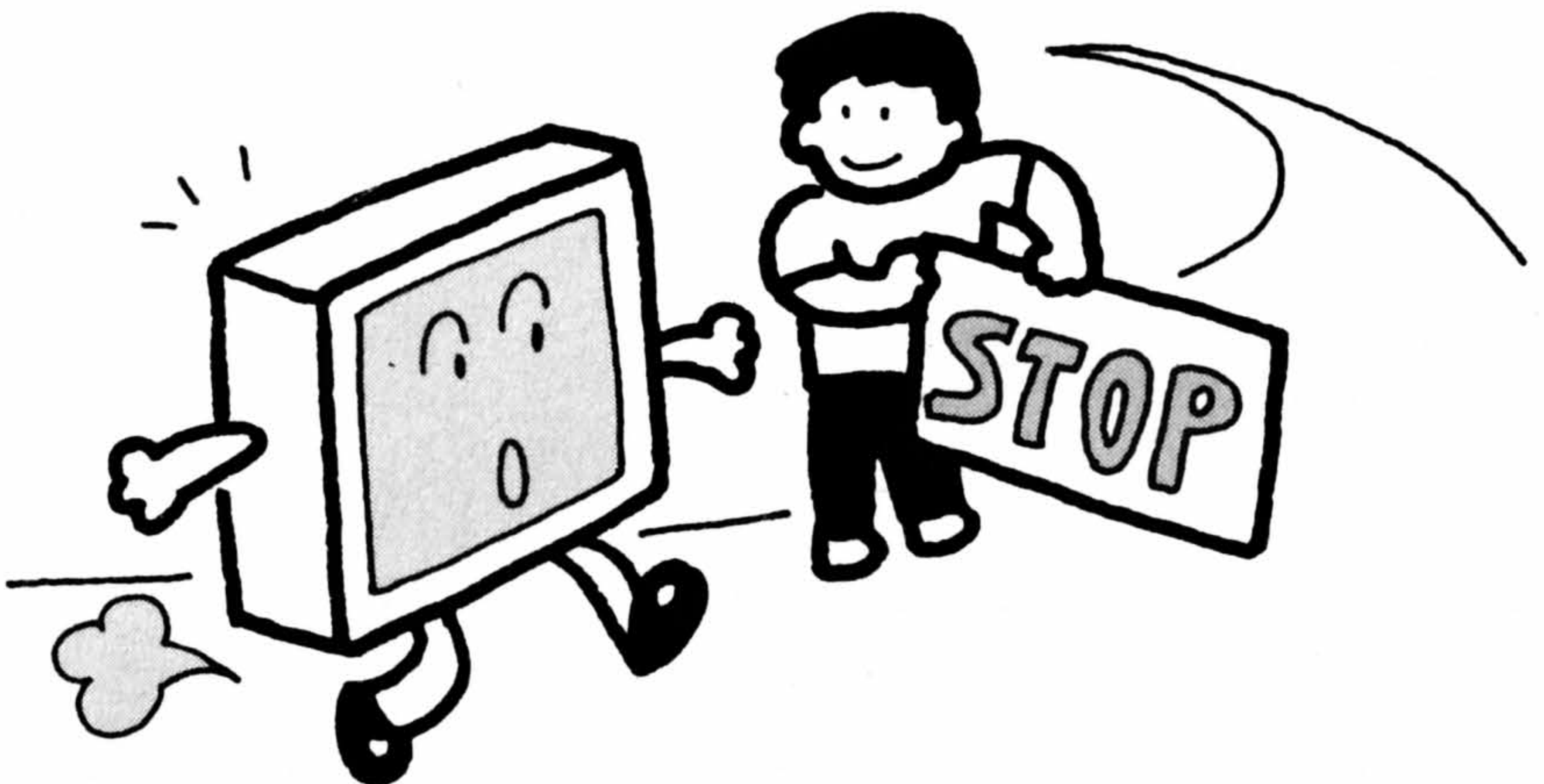
説明 プログラム中のどこで使用してもいい。STOP文を実行すると、"BREAK in 行番号"と表示してコマンドレベルに戻る。

停止したプログラムはCONT文で再実行できる。（CONTの項参照）また、STOPで停止したプログラムを終了させるときは、CTRL+STOPキーを同時に押す。

STOP命令は、プログラムの途中で各変数値を調べるときなどに使用する。

（例）

```
10 CLS
20 FOR I=1 TO 10
30 PRINT "#":NEXT I
40 STOP
50 FOR J=1 TO 10
60 PRINT "$":NEXT J
70 END
```

このプログラムを実行すると、#マークを表示して停止する。その後、CONT命令で再スタートさせると、画面に$マークが表示される。

END（エンド）

■プログラムを終了させる

プログラムの実行を終了させ、実行終了後はコマンドレベルに戻り、すべてのファイルを閉じる。

書式 END

説明　END文はプログラムを終了させたい場所のどこにでも、そして何個所に置いてもいい。また、プログラムの最後のEND文は省略してもいい。この場合にかぎりファイルは閉じない。

（例）
```
10 CLS
20 INPUT A
30 IF A=3 THEN END
40 GOTO 20
```

プログラムを実行後、1～10までの数字をキー入力してみる。30行でもし3なら、このプログラムは実行を終了し、コマンドレベルに戻る。このようにプログラムの途中で使用することができる。

READ～DATA（リード～データ）

■DATAを読み変数に代入

DATA文に含まれているデータを読み込み、READ文のすぐあとの変数に代入する。

書式
READ　変数，変数………
DATA　定数，定数………

説明　READ文とDATA文は組み合わせて使用される。READ文のあとには読み込み変数、DATAのあとには定数が書かれる。両方ともコンマで区切って複数の変数や定数を書くことができる。

READ文の変数は数値変数でも文字変数でもかまわない。READ文は行番号の小さいDATA文のデータから順番に1対1の対応で変数に代入してい

く。また、DATA文はプログラムのどの位置に置かれていても、READ文で読み込まれる。

（例）
```
10 READ I，N
20 PRINT I；N
30 DATA 2，3
40 END
```

```
RUN
 2  3
OK
■
```

このプログラムを実行すると、変数IにはDATA文の2が読み込まれ、変数NにはDATA文の3が順序よく読み込まれる。DATA文はREAD文の前に書かれていても結果は同じである。

注

1 READ文の変数は、数値変数でも文字変数でもいいが、数値変数なのにDATA文にA，B，Cなどと書くとエラーになる。もし、同じ型の変数でないときは読み込みができず、"Syntax error"と表示されて、プログラムが中断してしまう。

2 READ文の読み込み数とDATA文の個数とが一致しないときがある。DATA文の個数が多くとも、READ文で読み込むが、余分なデータは無視される。また、DATA文の個数が少ない場合には、読み込みができず、"Out of DATA"というエラーメッセージが表示され、プログラムが中断してコマンドレベルに戻る。

●サンプルプログラム

```
10 CLS
20 SCREEN2,2
30 FORN=1TO32
40 READ A
50 A$=A$+CHR$(A)
60 NEXT
70 SPRITE$(1)=A$
80 COLOR 2,1,1
90 PUT SPRITE1,(100,80),4,1
100 GOTO 100
110 REM DATA 32コ
120 DATA 0,0,1,3
130 DATA 3,7,15,63
140 DATA 224,63,15,7
150 DATA 3,3,1,0
160 DATA 0,128,0,1
170 DATA 2,6,6,255
180 DATA 255,255,6,6
190 DATA 2,1,0,128
```

RESTORE（レストア）

■ＲＥＡＤ文で読むＤＡＴＡ文を指定する

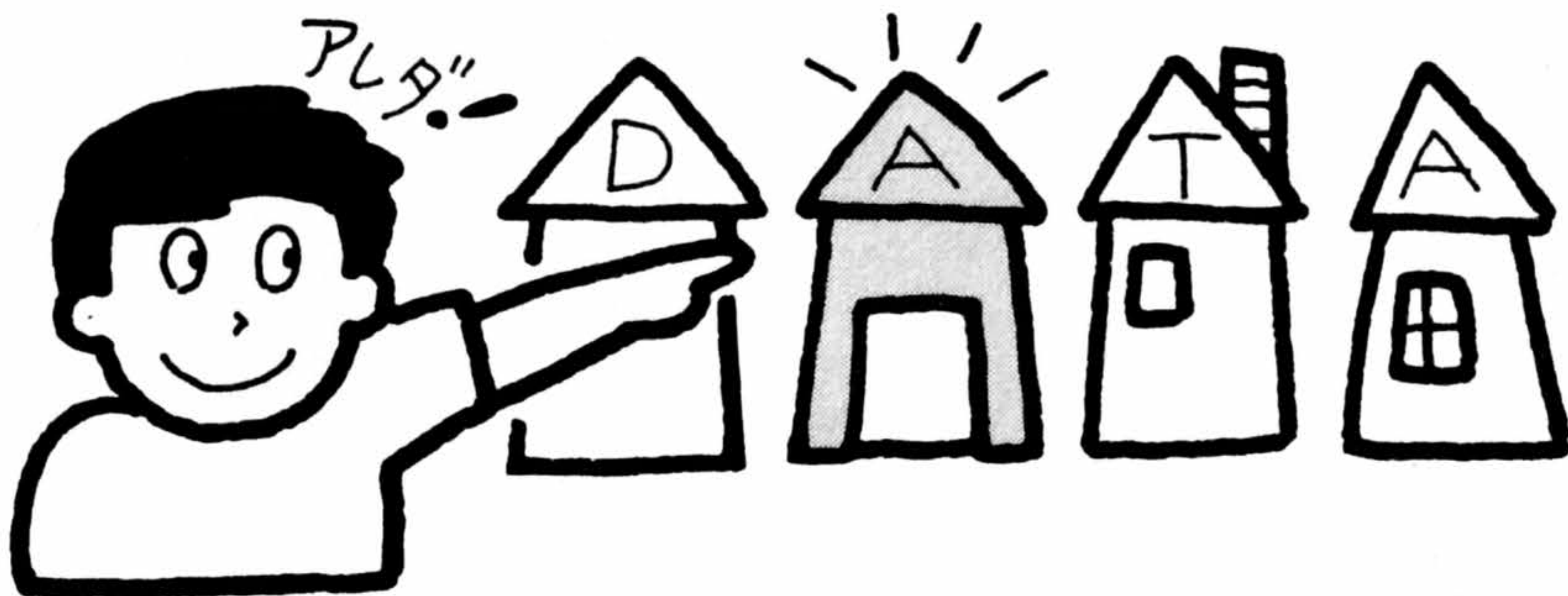
指定した行番号のＤＡＴＡ文から読み込む。何回も同じＤＡＴＡ文から読み込みたいときや、何回もＲＥＡＤ文で読み込むとき、また読み込みたいＤＡＴＡ文の定数を確実に読み込むときなどに使用する。

書式　ＲＥＳＴＯＲＥ　行番号

行番号を省略すれば、プログラムの最初のＤＡＴＡ文から読み始める。

書式例　ＲＥＳＴＯＲＥ　500

●サンプルプログラム

```
100 CLS
110 RESTORE 230
120 READ A
130 PRINTA;
140 IF A=1 THEN 160
150 GOTO120
160 RESTORE 220
170 READ A
180 PRINTA;
190 IF A=10 THEN 210
200 GOTO170
210 END
220 DATA 1,2,3,4,5,6,7,8,9,10
230 DATA 10,9,8,7,,5,4,3,2,1
```

ON ERROR GOTO (オン　エラー　ゴーツー)

エラー処理ルーチンへの開始行を指定

この命令を実行しておくと、エラーが発生したときに割り込みがかかり、指定したエラー処理ルーチンに強制的にジャンプする。

書式　ON　ERROR　GOTO　行番号

説明　通常、プログラムの実行時、エラーが発生すると、エラー表示が出て実行が中断されるが、この命令を実行しておくと、GOTOのあとで指定した行番号へジャンプしてプログラムが続行する。

行番号はエラー時の処理ルーチンが始まる行番号である。指定した行番号が存在しなければ、"Undefined line" と画面に表示され、プログラムは中断する。

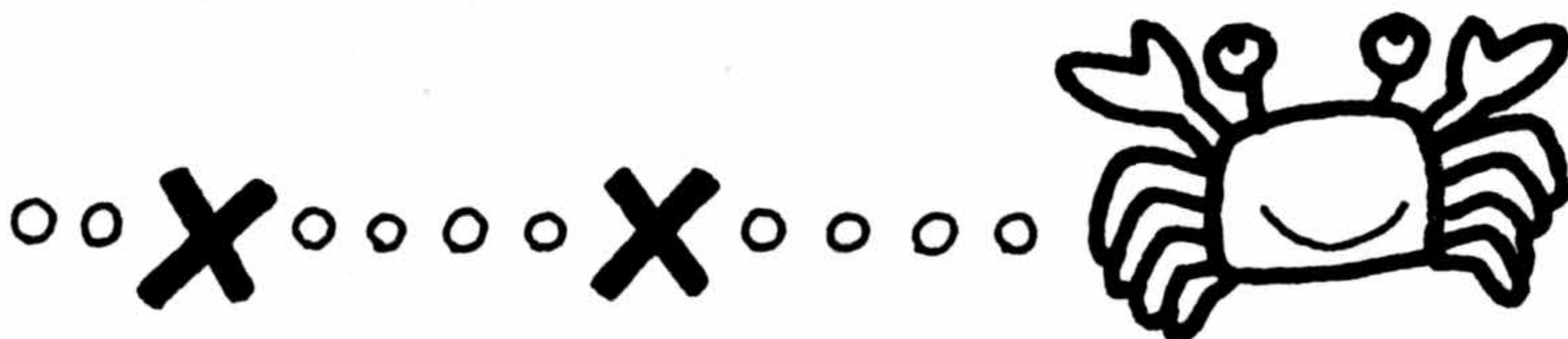
書式例　ON　ERROR　GOTO　50　…　エラーがあったら50行へ飛べ。

注

1　ON　ERROR　GOTOのあとの行番号を0にすると、それまでの定義は無効になり、BASICで用意してある通常のエラー処理が行われる。

ON　ERROR　GOTO文の効果は、つぎのON　ERROR　GOTOで定義するか、RUN命令またはCLEAR命令が実行されると無効になる。

2　エラー処理時のエラーコードやエラー発生時の行番号を知る関数がある。エラーコードを知るERR関数と行番号を知るERL関数である。

書式　A＝ERR
　　　　B＝ERL

ＳＰＲＩＴＥ＄（スプライトダラー）

■スプライトパターンを定義する

スプライト機能で使用するスプライトパターンを定義し、スプライトパターン番号を指定する。

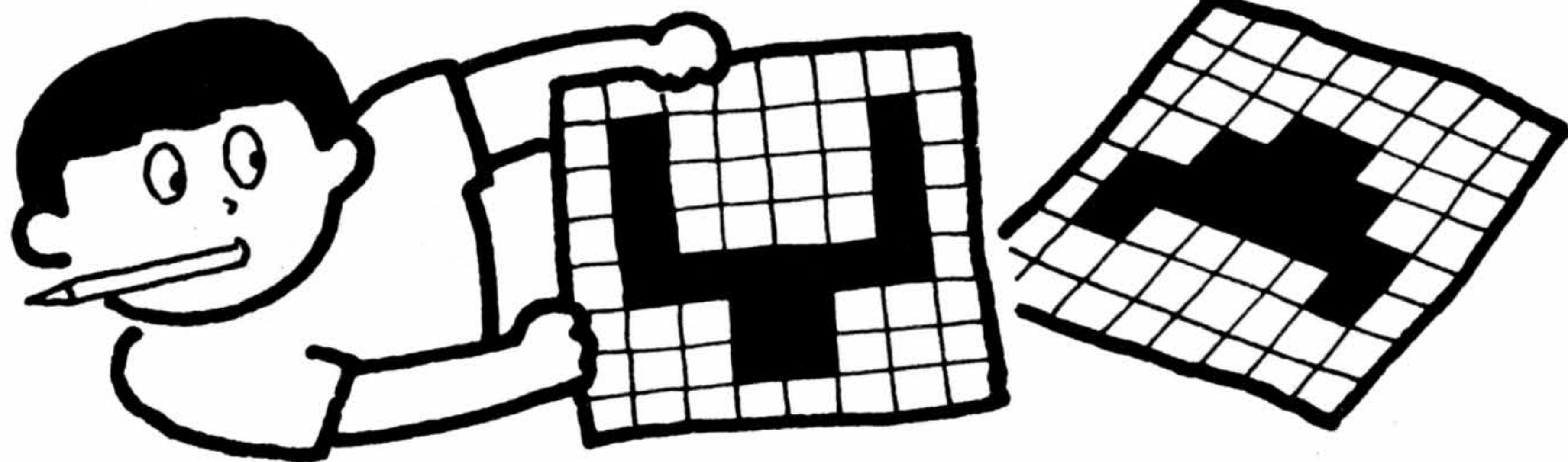
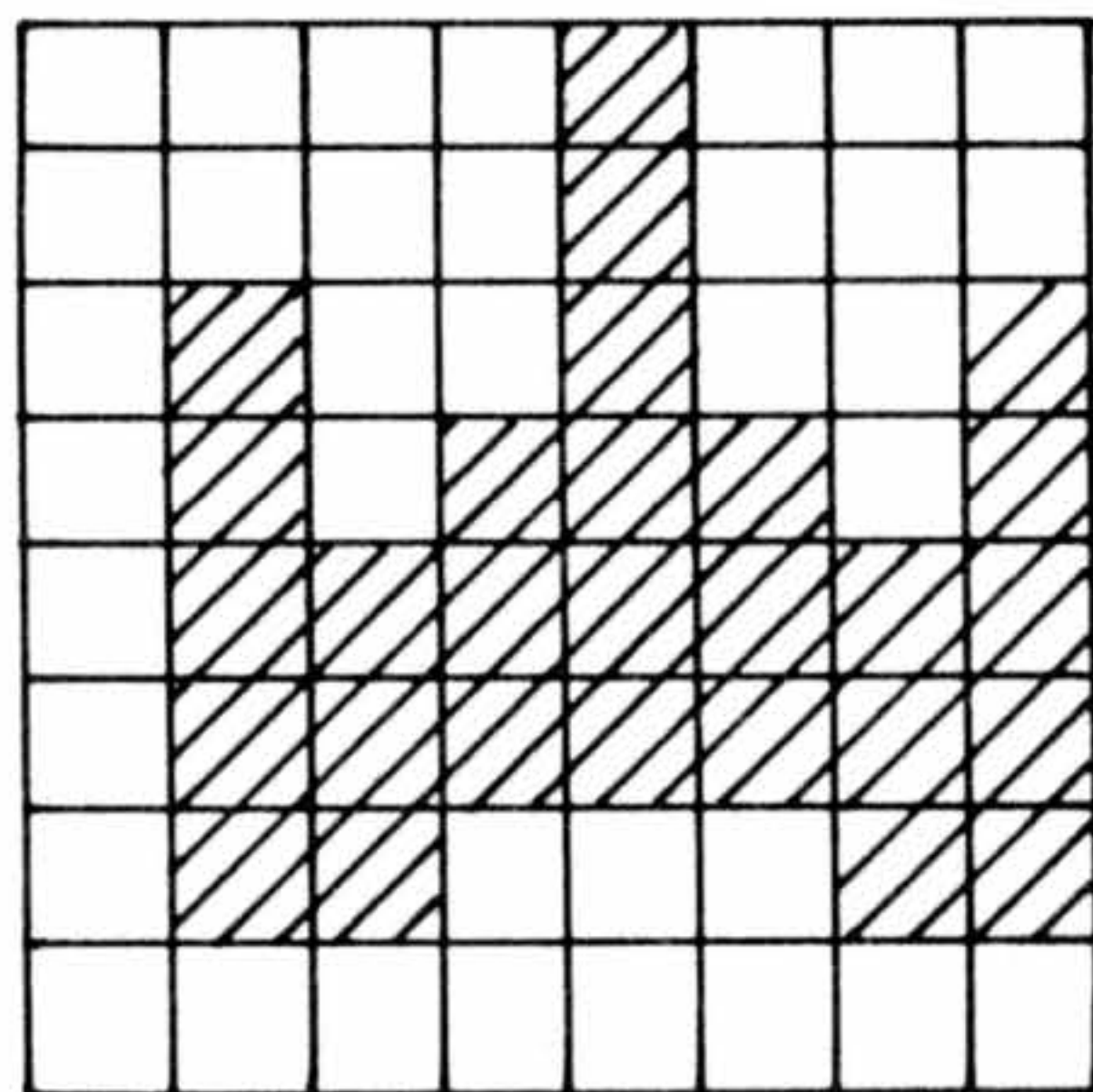
書式 ＳＰＲＩＴＥ＄（スプライトパターン番号）＝（スプライトパターンＤＡＴＡ）

説明 スプライトパターン番号で各パターンを定義する。番号の範囲は 0～255で 256個のスプライトパターンが定義できる。

ＳＣＲＥＥＮ文でスプライトサイズを指定した場合、サイズ０と１では 8×8ドットで256個、２と３では16×16ドットで 0～63の範囲で64個のスプライトパターンが定義できる。

スプライトパターンＤＡＴＡは 8× 8ドットの場合８文字分の文字列で、16×16ドットの場合は16文字分の文字列で定義する。

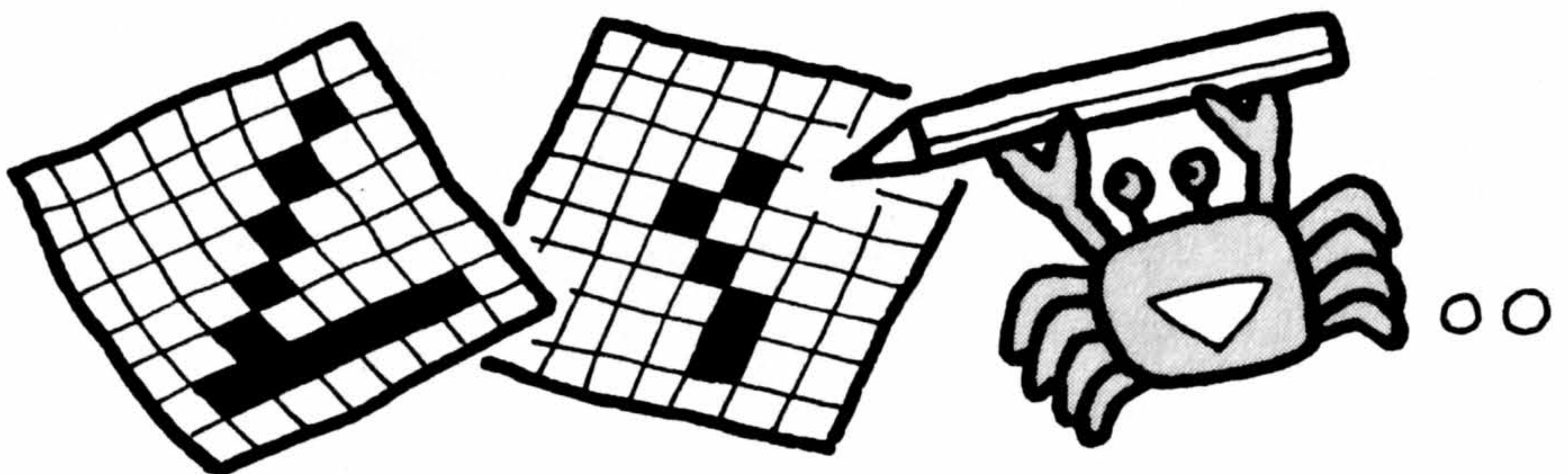
● 8×8ドットの定義方法

8×8のマスを書いて、好きな場所を塗りつぶしてみる。

マスを塗りつぶして絵が描けたら、横の１行ごとにキャラクタコードに直す。マスの白い部分を０，黒い部分を１とする。

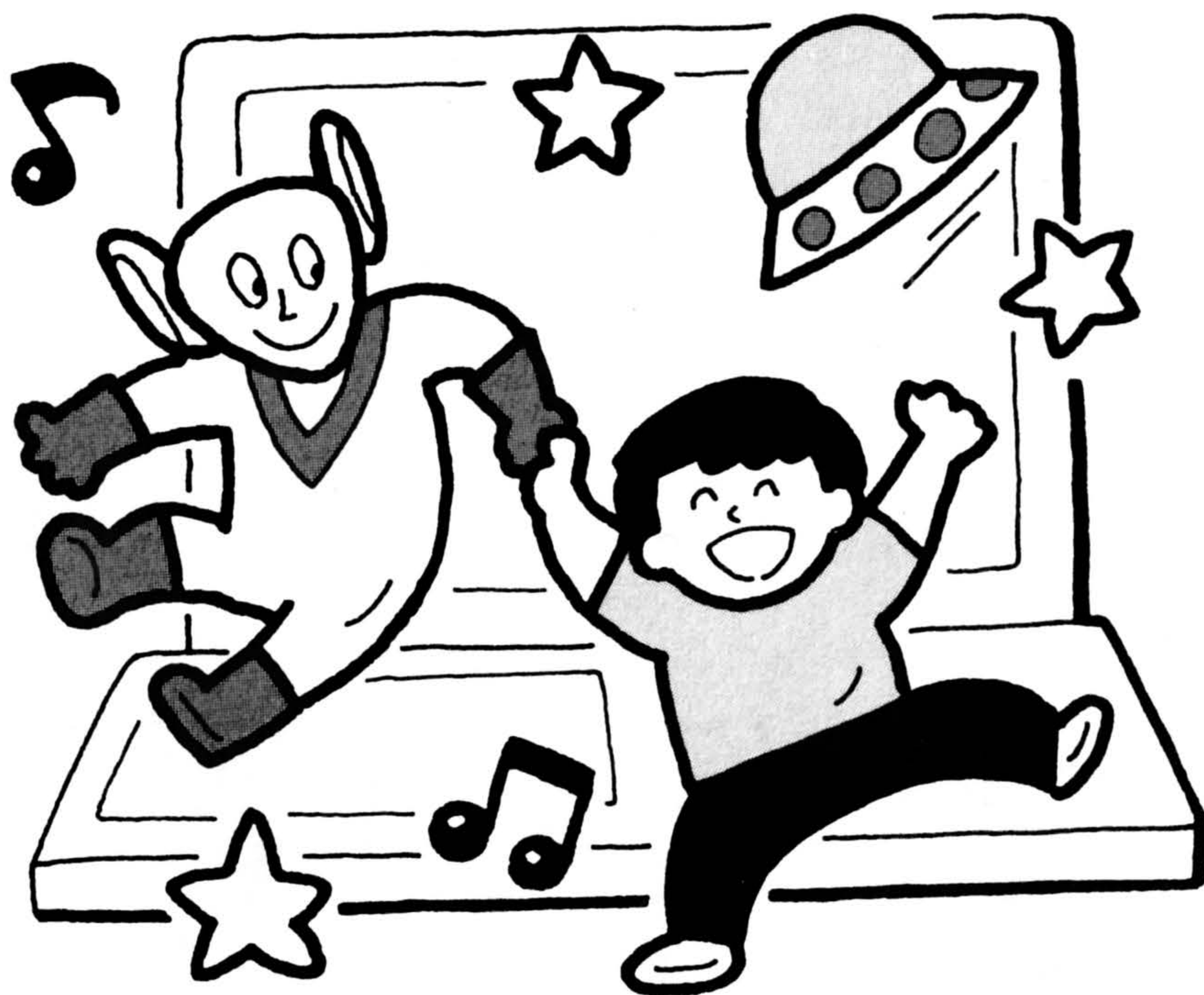
1. ０００01０００ ⟶ ＣＨＲ＄（＆Ｂ00001000）
2. ０００01０００ ⟶ ＣＨＲ＄（＆Ｂ00001000）
3. ０1００1００1 ⟶ ＣＨＲ＄（＆Ｂ01001001）
4. ０1０111０1 ⟶ ＣＨＲ＄（＆Ｂ01011101）
5. ０1111111 ⟶ ＣＨＲ＄（＆Ｂ01111111）
6. ０1111111 ⟶ ＣＨＲ＄（＆Ｂ01111111）
7. ０11０００11 ⟶ ＣＨＲ＄（＆Ｂ01100011）
8. ００００００００ ⟶ ＣＨＲ＄（＆Ｂ00000000）

０と１で８桁のマス目が埋ったら、それにＣＨＲ＄を使用して、矢印の右のように書く。これで実際にパターンを定義してみる。

```
10 Ａ１＄＝ＣＨＲ＄（＆Ｂ00001000）
20 Ａ２＄＝ＣＨＲ＄（＆Ｂ00001000）
30 Ａ３＄＝ＣＨＲ＄（＆Ｂ01001001）
40 Ａ４＄＝ＣＨＲ＄(＆Ｂ01011101)
50 Ａ５＄＝ＣＨＲ＄（＆Ｂ01111111）
60 Ａ６＄＝ＣＨＲ＄(＆Ｂ01111111)
70 Ａ７＄＝ＣＨＲ＄（＆Ｂ01100011）
80 Ａ８＄＝ＣＨＲ＄(＆Ｂ00000000)
90 ＳＰＲＩＴＥ＄（０）＝Ａ１＄＋Ａ２＄＋Ａ３＄＋Ａ４＄＋Ａ５＄＋
                       Ａ６＄＋Ａ７＄＋Ａ８＄
```

これでスプライトパターンの（０）に定義されたわけである。

このパターンを使用したプログラムの例である。行番号10〜90までにつづけて、下の100 〜120 行を入力する。

```
100  SCREEN 1
110  PUT SPRITE 0,(150,50),1,0
120  GOTO 120
```

追加したプログラムを実行すると、スプライトパターンで定義したキャノンらしきものが黒い色で表示される。スプライトパターンを表示させるにはグラフィックモードで実行するのだが、プログラムが終わるとすぐ画面の表示が消えてしまうので、プログラム中 120行で無限ループにして消えないようにしている。プログラムを終了したい場合は、 CTRL + STOP キーを押して、テキストモードに戻す。

PUT SPRITE(プット スプライト)

スプライトパターンを表示する

指定したスプライトをスプライト面の指定位置に表示させる。

書式 PUT SPRITE スプライト面番号, x, y, カラーコード, スプライトパターン番号

説明 SPRITE$(n)で代入された文字型変数のパターンを、指定したスプライト番号のスプライト面の座標位置(x, y)の位置に表示する。

SCREENでのスプライトサイズ指定はつぎのようになっている。

(スプライトサイズ)

0	8×8	ドット
1	8×8	ドット拡大
2	16×16	ドット
3	16×16	ドット拡大

スプライト番号は 0〜31までの値で指定できる。1スプライトの面には1つのスプライトパターンだけを表示する。2つ以上のスプライトパターンを表示しようとすると、以前に表示したスプライトパターンは消去されてしまう。

スプライト面は同じように32まで使用できるが、水平方向に4個までのスプライトパターンだけが表示できる。

座標(x, y)は、x座標が−32〜255 で、y座標が−32〜191 の範囲の数式変数および数値で指定する。

カラーコードは、スプライトパターンの色を指定し、スプライト番号はあらかじめSPRITE$で定義しておいたスプライトパターンの番号である。

書式例 PUT SPRITE 0,(100 , 100), 8, 0

スプライト0面にSPRITE$で定義したスプライトパターン0番をx座標100 , Y座標 100の指定場所にカラー8(赤)の色をつけて表示させる。

(例)
```
10 SCREEN 2, 2
20 SPRITE$(0)=STRING$ (32,CHR$ (&HF
3))
```

```
30 PUT SPRITE 0, (100 , 10), 8, 0
40 PUT SPRITE 1, (100 , 90), 12, 0
50 GOTO 50
```

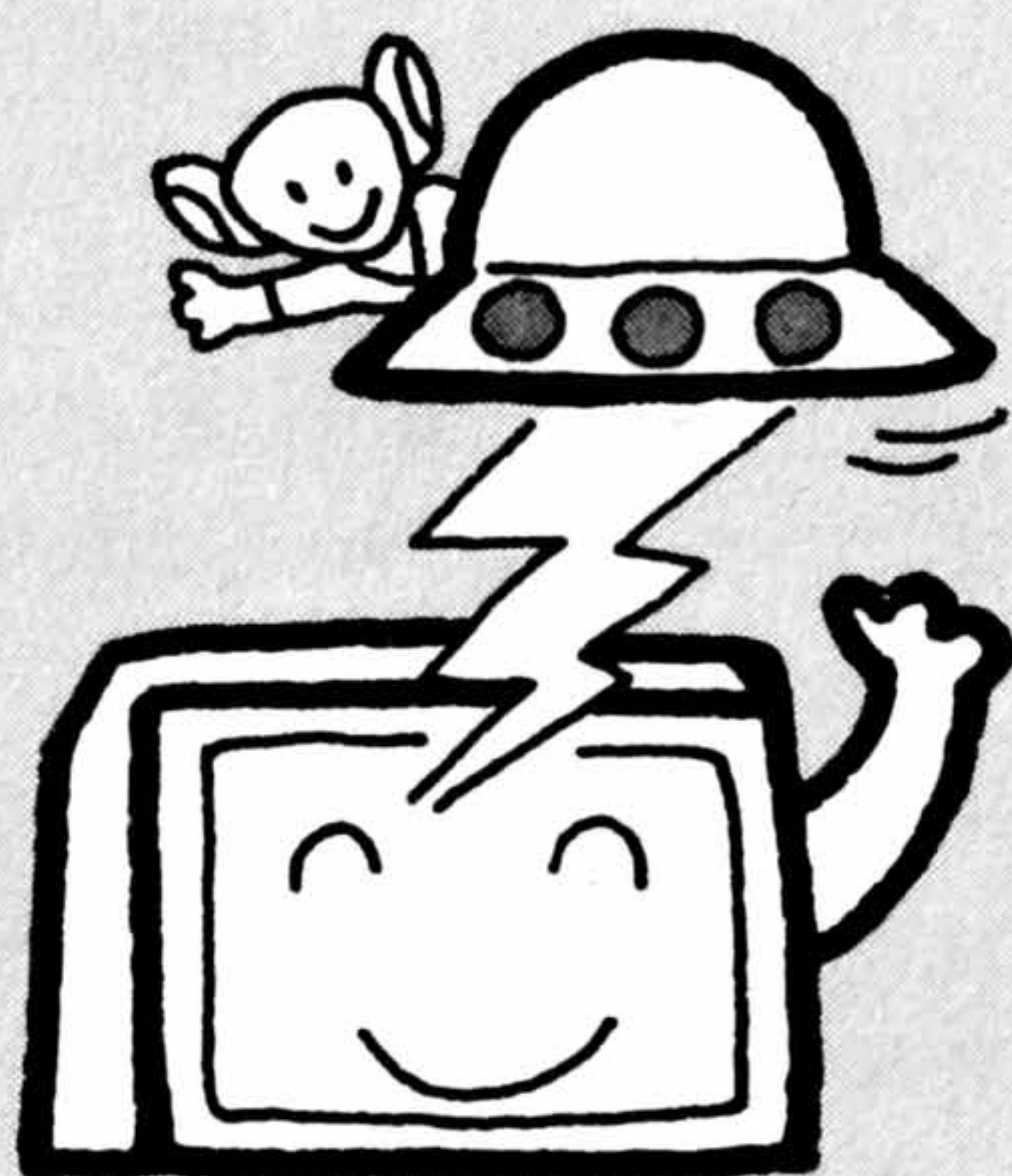
10行のSCREEN2，2を2，3に変えるとスプライト画面は大きくなる。

注

1　PUT SPRITE文の各座標を省略すればスプライト面の現在の値が用いられ、カラーコードを省略すると前景色が用いられる。

2　Y座標に 208を指定すれば、そのスプライト面以降のスプライトパターンはすべて画面から消去する。また、209 を指定すると指定したスプライト面に表示されていたスプライトパターンは消去してしまう。

●サンプルプログラム

```
100 CLS
110 A1$=CHR$(&B00011000)
120 A2$=CHR$(&B00100100)
130 A3$=CHR$(&B01000010)
140 A4$=CHR$(&B10010101)
150 A5$=CHR$(&B10010101)
160 A6$=CHR$(&B01000010)
170 A7$=CHR$(&B00100100)
180 A8$=CHR$(&B00011000)
190 B1$=CHR$(&B01111110)
200 B2$=CHR$(&B10000001)
210 B3$=CHR$(&B10000001)
220 B4$=CHR$(&B10100101)
230 B5$=CHR$(&B10100101)
240 B6$=CHR$(&B10000001)
250 B7$=CHR$(&B10000001)
260 B8$=CHR$(&B01111110)
270 SPRITE$(0)=A1$+A2$+A3$+A4$+A5$+A6$+A7$+A8$
280 SPRITE$(1)=B1$+B2$+B3$+B4$+B5$+B6$+B7$+B8$
290 SCREEN 2
300 PUT SPRITE0,(50,60),2,0
310 PUT SPRITE1,(100,130),8,1
320 GOTO320
```

④ グラフィックコマンド

PSET（ポイントセット）

■描きたい場所に点を描く

グラフィック画面の指定した座標位置に、指定したカラーコードの色で点を描く。

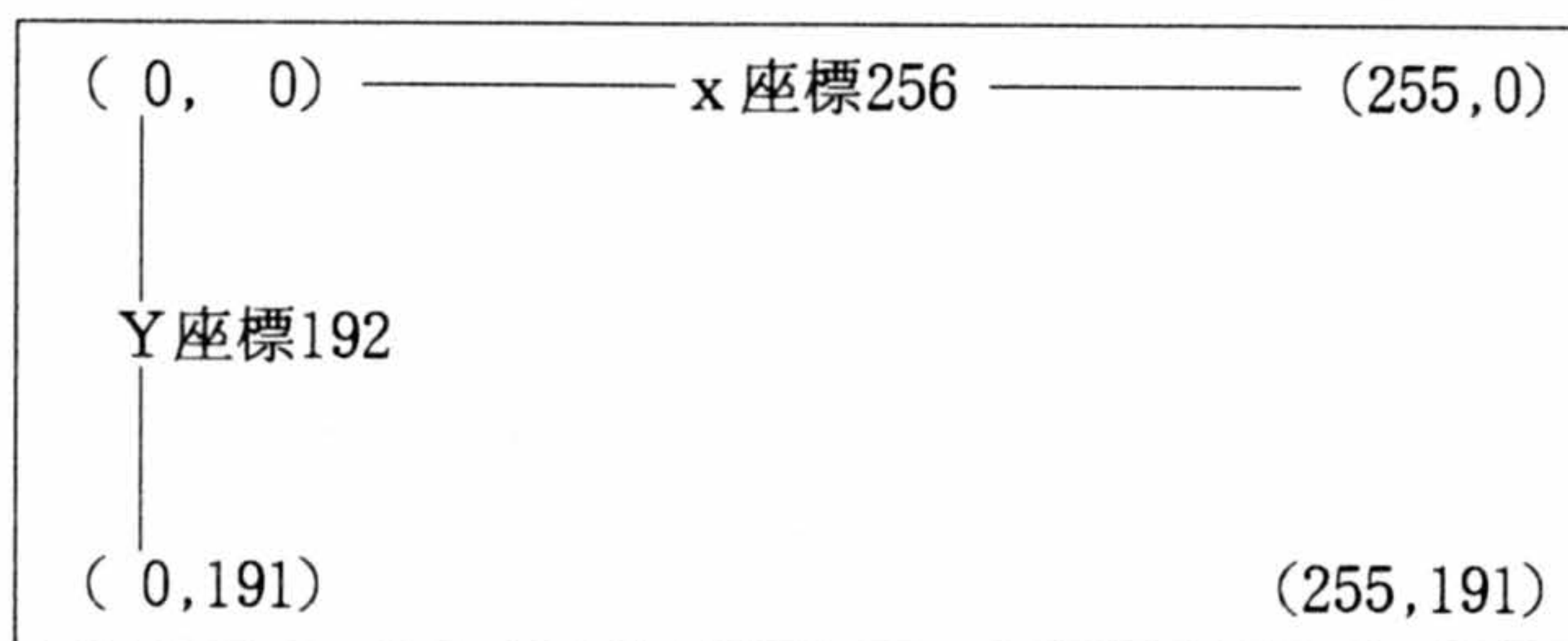
書式 PSET　(x, y), カラーコード

説明 座標（x，y）の位置に指定したカラーコードの色で点を描く。カラーコードを省略すれば、COLOR命令で指定した表示色が使用される。

また、（x，y）の前にSTEPをつけてPSET　STEP（x，y）とすると、一番最近表示された座標位置からの相対位置となる。

つまり、一番最近表示された座標が（100，100）で、PSET　STEP（10，−10），2を実行すれば、x座標は100＋10で110，y座標は100−10で90となり、座標（110，90）の位置に緑色（カラーコード2）の点を表示する。

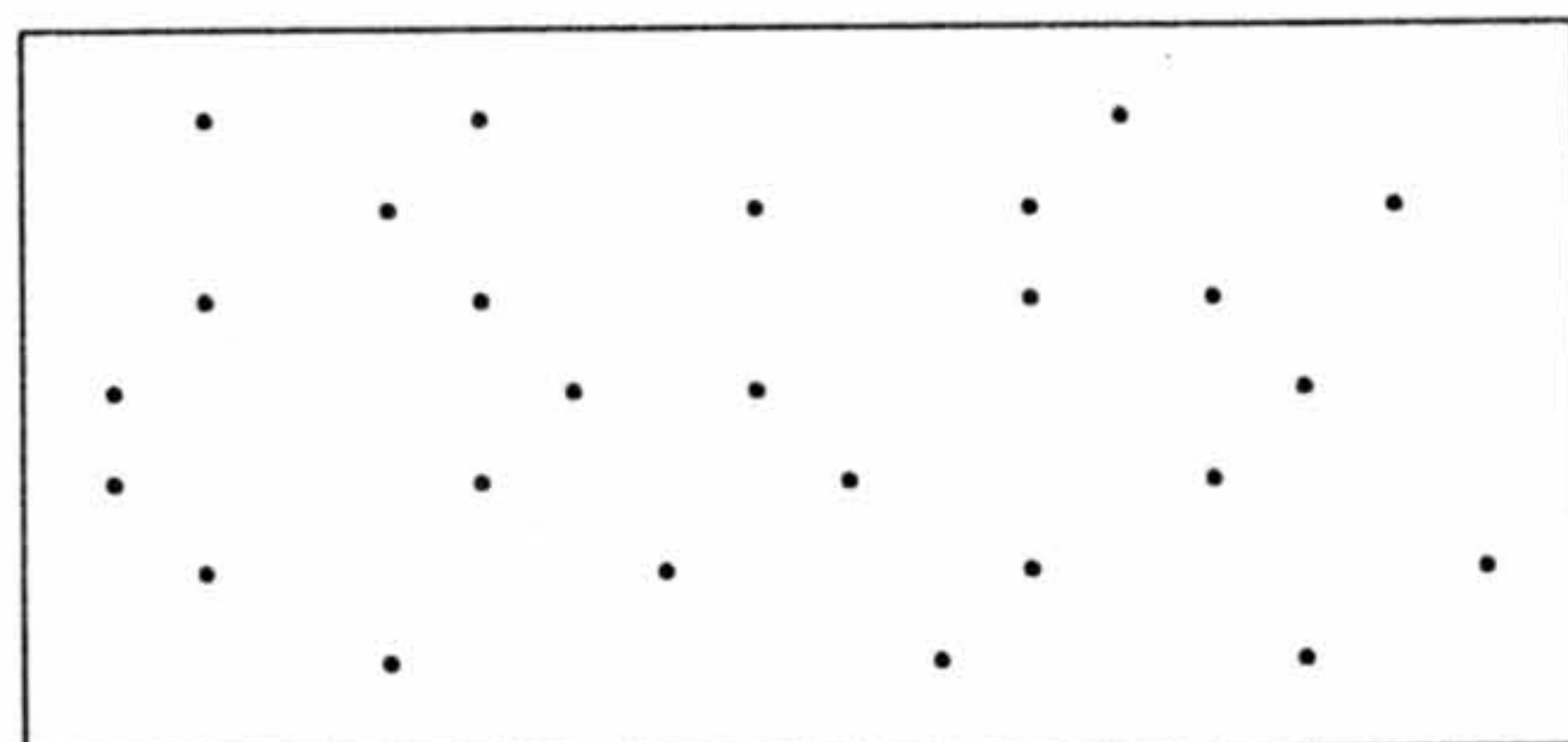
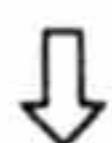
（x座標とy座標）

```
( 0,  0) ────── x座標256 ────── (255,0)
 |
 |
Y座標192
 |
 |
( 0,191)                        (255,191)
```

書式例 PSET　(10, 50), 10

座標（10，50）の位置に黄色い点を描く。

PSET　STEP (20, 20)

一番最近表示された座標から下に20ドット、右に20ドットずれた位置にCOLOR命令で指定した表示色の色で点を描く。

（例）
```
10 SCREEN  2
20 X=INT (RND (1) *255 )
30 Y=INT (RND (1) *191 )
40 C=INT (RND (1) *16  )
50 PSET  (X, Y), C
60 GOTO 20
```
⇩

プログラムをＲＵＮさせると、いろいろな色の点が不規則に表示されていく。これは、20行でｘ軸の座標値、30行でｙ軸の座標値、40行でカラーコードの値をそれぞれ乱数で発生させ、50行のＰＳＥＴ命令で表示させているからである。

●サンプルプログラム

```
100 SCREEN 2
110 FOR N=1TO290
120 PSET(N,10),2
130 PSET(N,20),8
140 PSET(N,30),4
150 PSET(N,40),12
160 PSET(N,50),9
170 PSET(N,60),5
180 PSET(N,70),3
190 PSET(N,80),13
200 PSET(N,90),7
210 PSET(N,100),9
220 PSET(N,110),10
230 PSET(N,120),2
240 PSET(N,130),8
250 PSET(N,140),5
260 PSET(N,150),12
270 PSET(N,160),3
280 PSET(N,170),7
290 PSET(N,180),14
300 NEXT
310 GOTO310
```

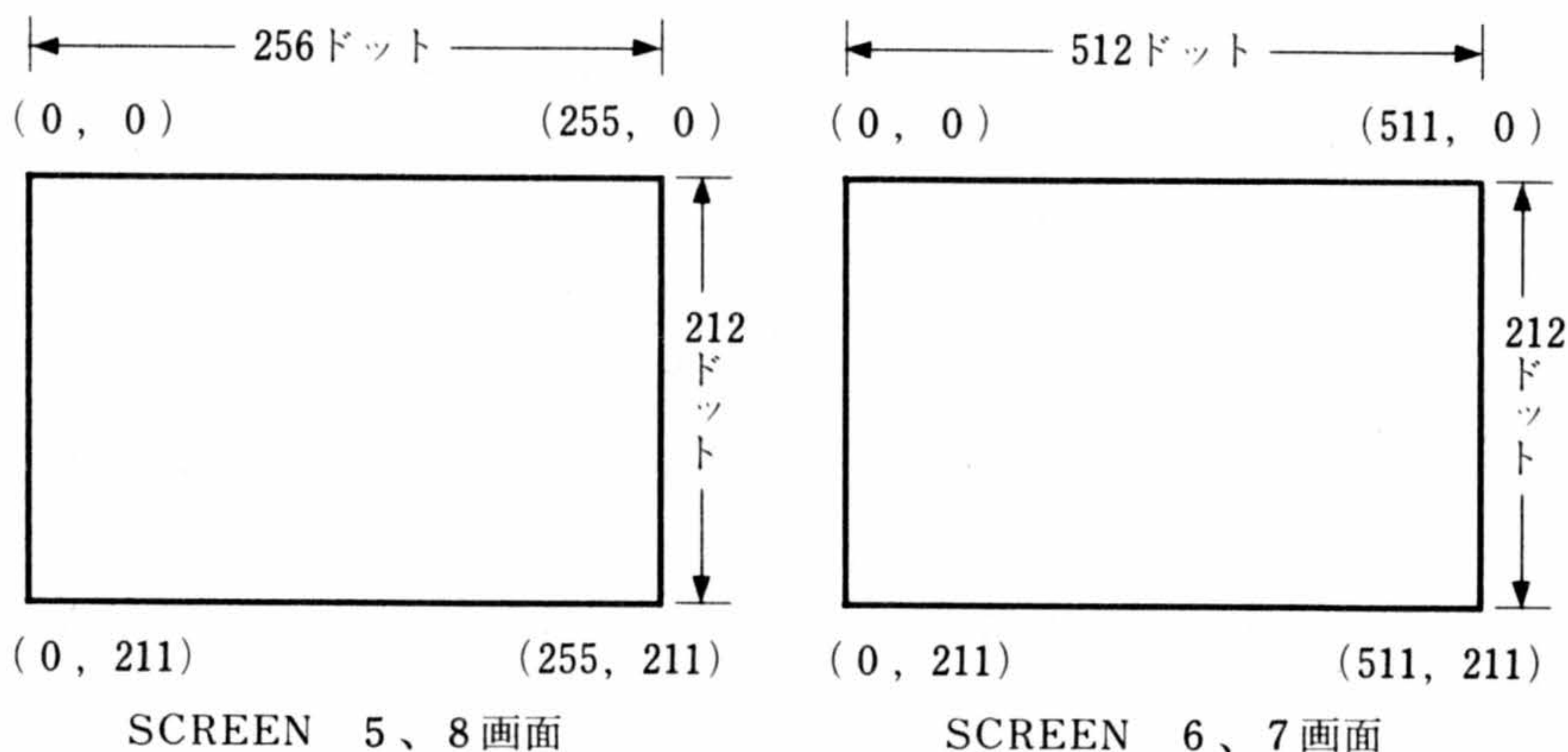
注

1　この命令はグラフィックモードのときだけ使用できる。

2　ＳＣＲＥＥＮ　２の高解像のグラフィックモードのとき、すべてのドットに対して自由に色をつけることはできない。

3　SCREEN　5～8では、カラーコードのあとに論理演算指定を付け、論理演算ができる。

256ドット
(0, 0)　(255, 0)
212ドット
(0, 211)　(255, 211)
SCREEN　5、8画面

512ドット
(0, 0)　(511, 0)
212ドット
(0, 211)　(511, 211)
SCREEN　6、7画面

PRESET（ポイントリセット）

■指定した位置の点の表示を消す

グラフィック画面に表示してある点の座標位置を指定し、その点を指定したカラーコードの色で消す。

書式　PRESET　(x, y), カラーコード

説明　座標（x，y）の位置に表示してある点をカラーコードの色で消す。カラーコードを省略すれば、COLOR命令で指定した背景色が使われる。（PSETの項を参照）

書式例　PRESET　(100, 100), 8

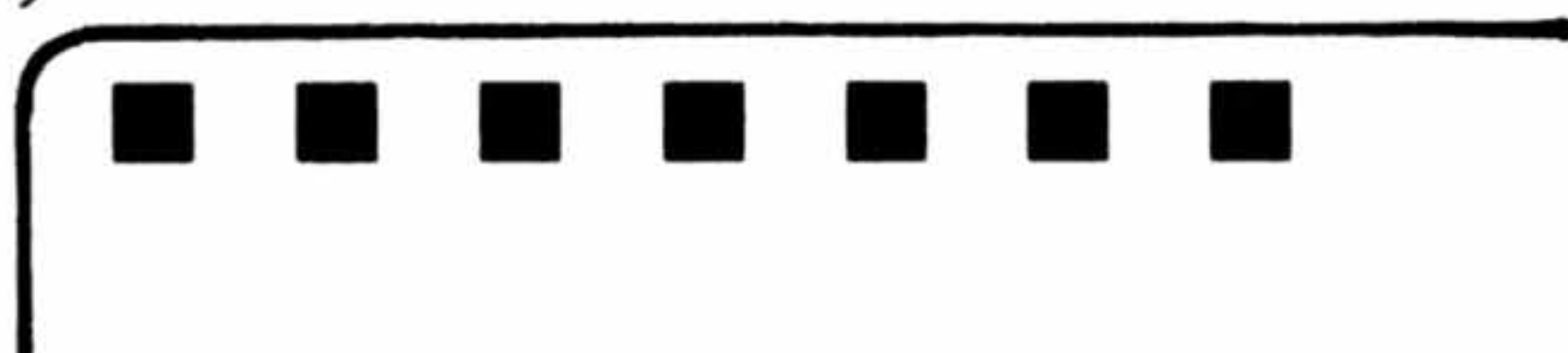
座標（100, 100）の位置に表示してある点を赤い点で消す。

PRESET　(50, 80)

座標（50, 80）の位置に表示してある点をCOLOR命令で指定した背景色で消す。

（例）
```
10 SCREEN 3
20 COLOR 15, 4, 7
```

```
30 LINE  (0, 100) - (250 , 100)
40 FOR  I = 4  TO  244  STEP  8
50 PRESET  (I, 100)
60 NEXT
70 GOTO 70
```

■ ■ ■ ■ ■ ■ ■

上のプログラムは、30行のＬＩＮＥ命令で表示した線を、40〜60行のＰＲＥＳＥＴを使ったプログラムで、一定間隔の点を消し、直線を点線にする。

注

1　この命令は、グラフィックモードのときだけ使用できる。

2　（ｘ，ｙ）の前にＳＴＥＰを付けてＰＲＥＳＥＴ　ＳＴＥＰ（ｘ，ｙ）とすれば、一番最近表示された座標位置からの相対位置となる。（ＰＳＥＴの項を参照）

3　ＳＣＲＥＥＮ　５〜８では、カラーコードのあとに論理演算指定を付け、論理演算ができる。

ＬＩＮＥ（ライン）

■２点間に線を引いたり箱を描いたり

グラフィック画面の上で指定した２点間を直線で結んだり、指定した２点を結ぶ線を対角線とする箱を表示したりする。

書式　LINE　(x 1,y 1) − (x 2,y 2), カラーコード, B（またはBF）

説明　座標（ｘ１，ｙ１）と座標（ｘ２，ｙ２）の２点間を結ぶ直線を、指定したカラーコードの色で描く。最後にＢを加えれば指定した２点間を対角線とする四角形を描く。Ｂの代わりにＢＦとすれば、指定した２点間を対角線とする四角形を描き、その内側を指定したカラーコードの色で塗りつぶす。

カラーコードを省略したときは、ＣＯＬＯＲ命令で指定した表示色が使用される。

書式例　LINE　(10, 10) − (200 , 100), 8

座標（10，10）と座標（200 ，100）を赤い線で結ぶ。

LINE　(20, 30) − (150 , 120),, B

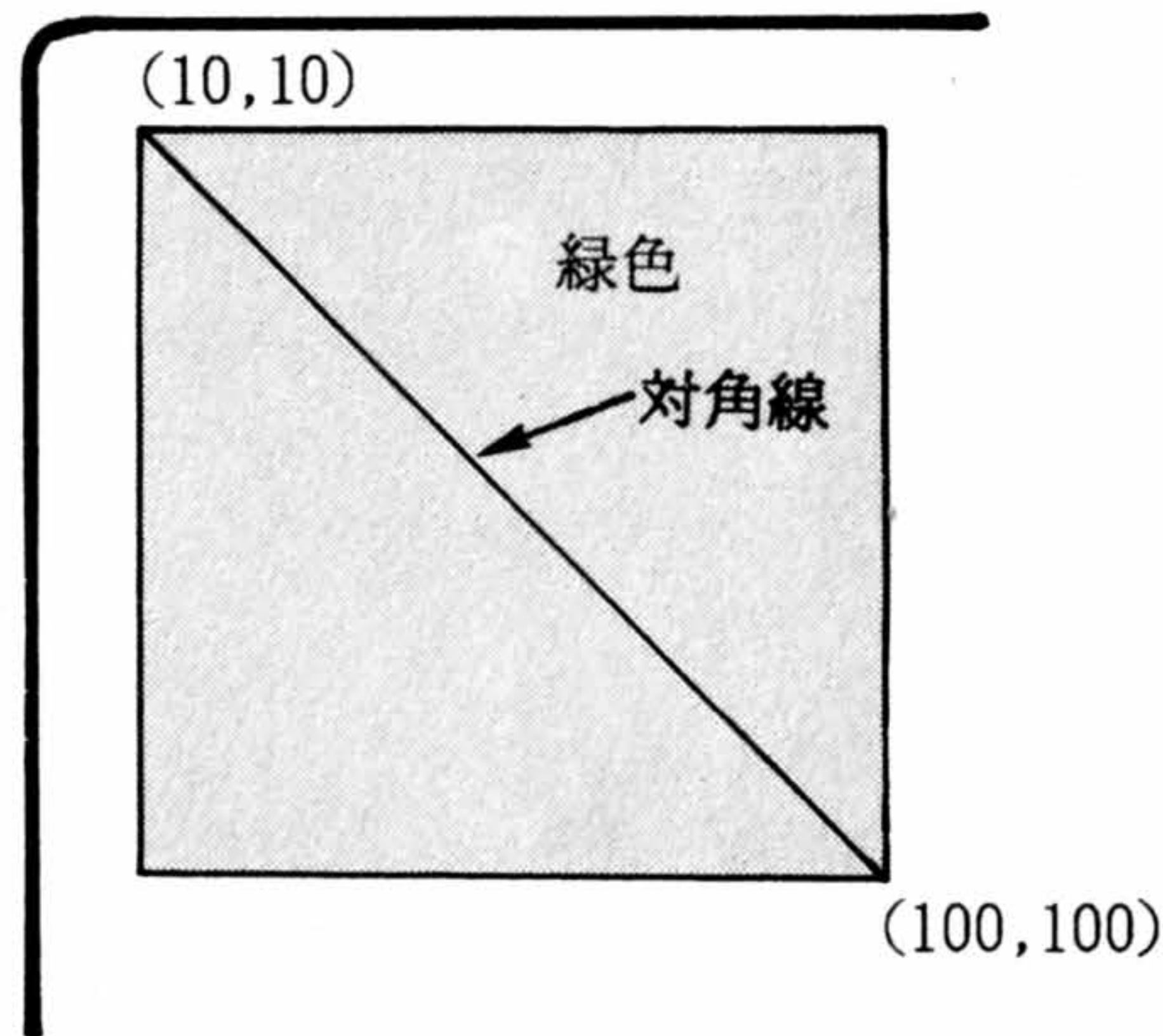
座標（20，30）と座標（150 ，120）を対角線とする四角形を表示する。色はＣＯＬＯＲ命令で表示した表示色を使う。

ＬＩＮＥ　(10，50) − (170 ，180)，8，ＢＦ

座標（10，50）と座標（170 ，180）を対角線とする四角形を表示して、その内側を赤で塗りつぶす。

```
（例）  10 SCREEN 2
        20 LINE (10,10)-(100,100),2,B
        30 GOTO 30
```

上のプログラムは、座標（10，10）と座標（100 ，100 ）を対角線とする四角形を表示する。

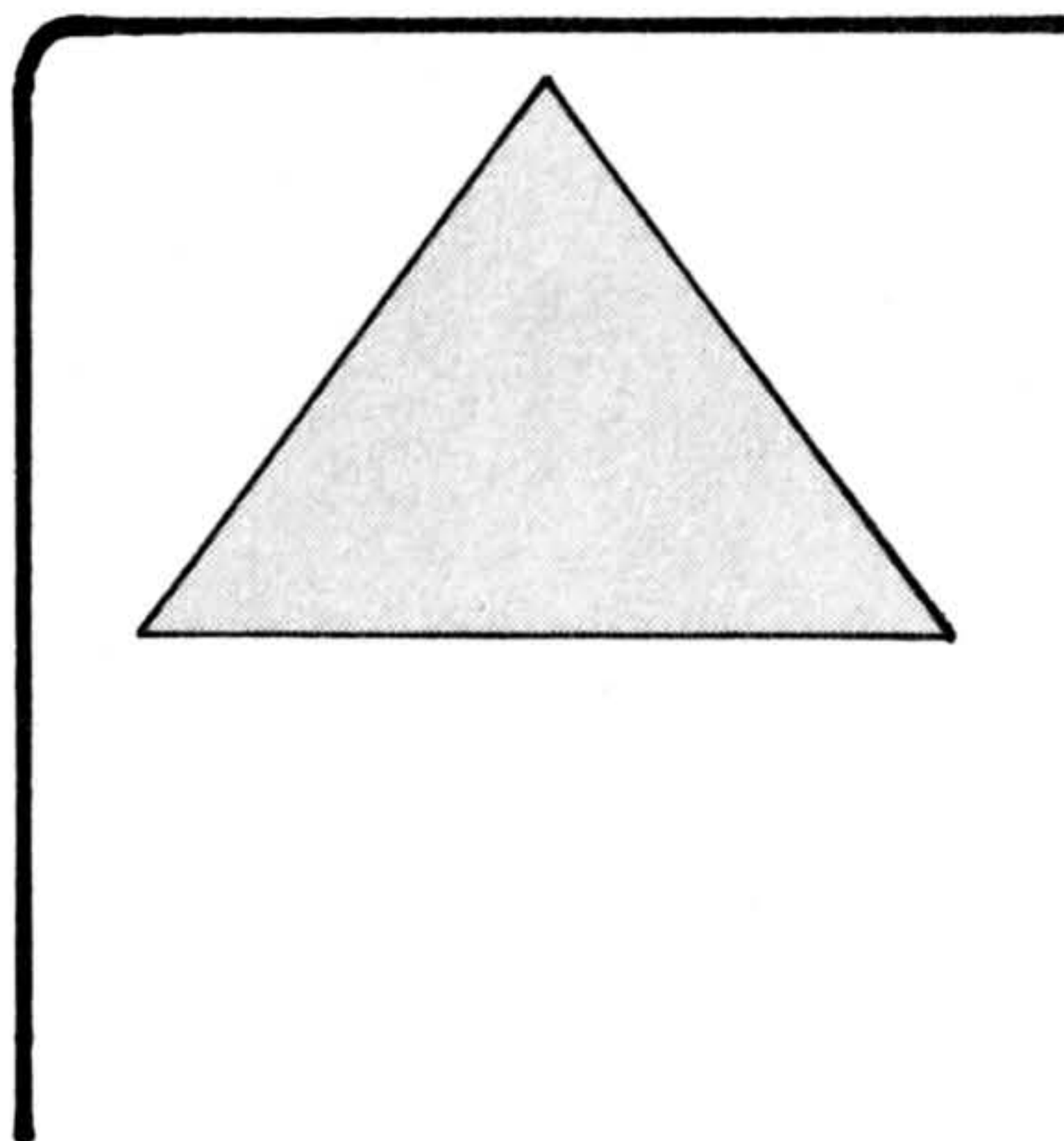
20行の末尾のＢをＢＦとすれば、四角の中を緑色に塗りつぶす。

●３点間以上を直線で結ぶ

３点以上を直線で結ぶときは、直線の数だけＬＩＮＥ命令を使用するが、２つめからのＬＩＮＥ文は、（ｘ１，ｙ１）を省略することができる。

たとえば三角形を書くときは、つぎのようになる。

```
10 SCREEN 2
20 LINE (100,10)-(20,90),15
30 LINE -(180,90),15
40 LINE -(20,90),15
```

```
50 GOTO 50
```

注

1　この命令はグラフィックモードのときだけ使用できる。

2　（x，y）の前にＳＴＥＰをつけて、ＬＩＮＥ ＳＴＥＰ（x 1，y 1）－ＳＴＥＰ(x 2，y 2)としたときは、一番最近表示された座標位置からの相対距離となる。

●サンプルプログラム

```
100 SCREEN2
110 X=10:Y=10:XX=240:YY=180:A=2
120 FOR N=10 TO 100 STEP 7
130 LINE(X+N,Y+N)-(XX-N,YY-N),A,BF
140 A=A+1:IF A=15 THENA=2
150 NEXT
160 GOTO160
```

ＣＩＲＣＬＥ（サークル）

円やだ円、扇形を描く

グラフィック画面の任意の点を中心に円や円弧、扇形、だ円などを表示する。

書式 ＣＩＲＣＬＥ　（x，y），半径，カラーコード，開始角度，終了角度，比率

説明 ＣＩＲＣＬＥ　（x，y），半径は座標（x，y）を中心に、指定された半径の大きさの円を表示する。カラーコードは、描く円の色を指定する。カラーコードを省略すると、ＣＯＬＯＲ命令で指定した表示色になる。

開始角と終了角を指定すると、指定した角度の範囲の円弧を表示する。指定する角度の単位はラジアンで、０～２π（パイ）の範囲である。この値を負にすると、円弧の両端と円の中心が線で結ばれて扇形となる。

開始角を省略すれば０、終了角を省略すれば２πが代入されて円となる。したがって円を表示するときは、指定しなくていい。

比率を指定すると、だ円が表示できる。比率＝垂直方向の半径／水平方向の半径である。比率が１より小さいときは、横長のだ円、１より大きいときは縦長のだ円になる。

だ円の場合半径は、垂直方向と水平方向のどちらか大きいほうの半径を指定する。

書式例 ＣＩＲＣＬＥ　(100，100)，50，２

座標（100，100）を中心に、半径50の緑色の円を表示する。

ＣＩＲＣＬＥ　(100，100)，50，，０，π

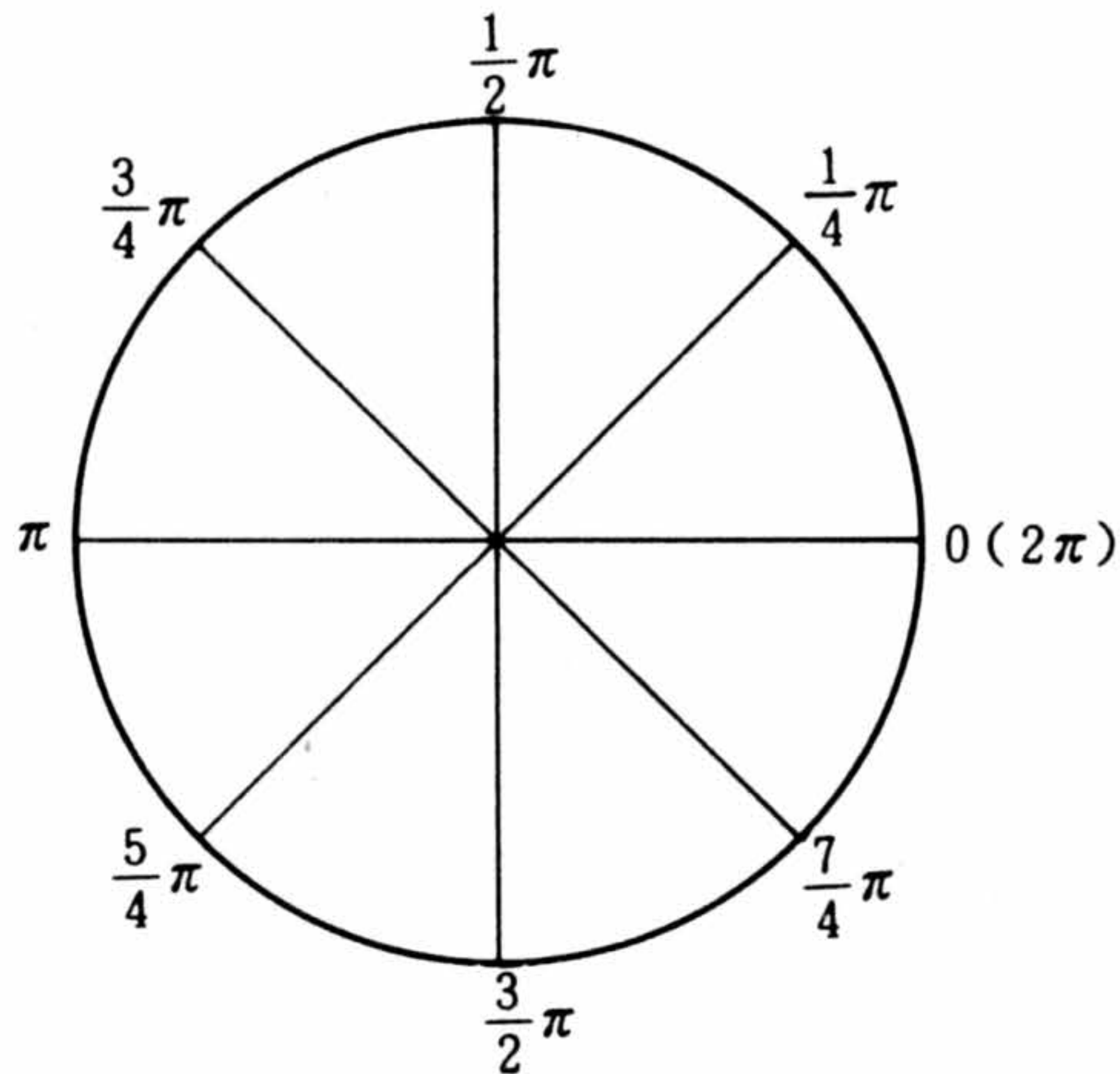
座標（100，100）を中心に、半径50の上半分の円弧を表示する。

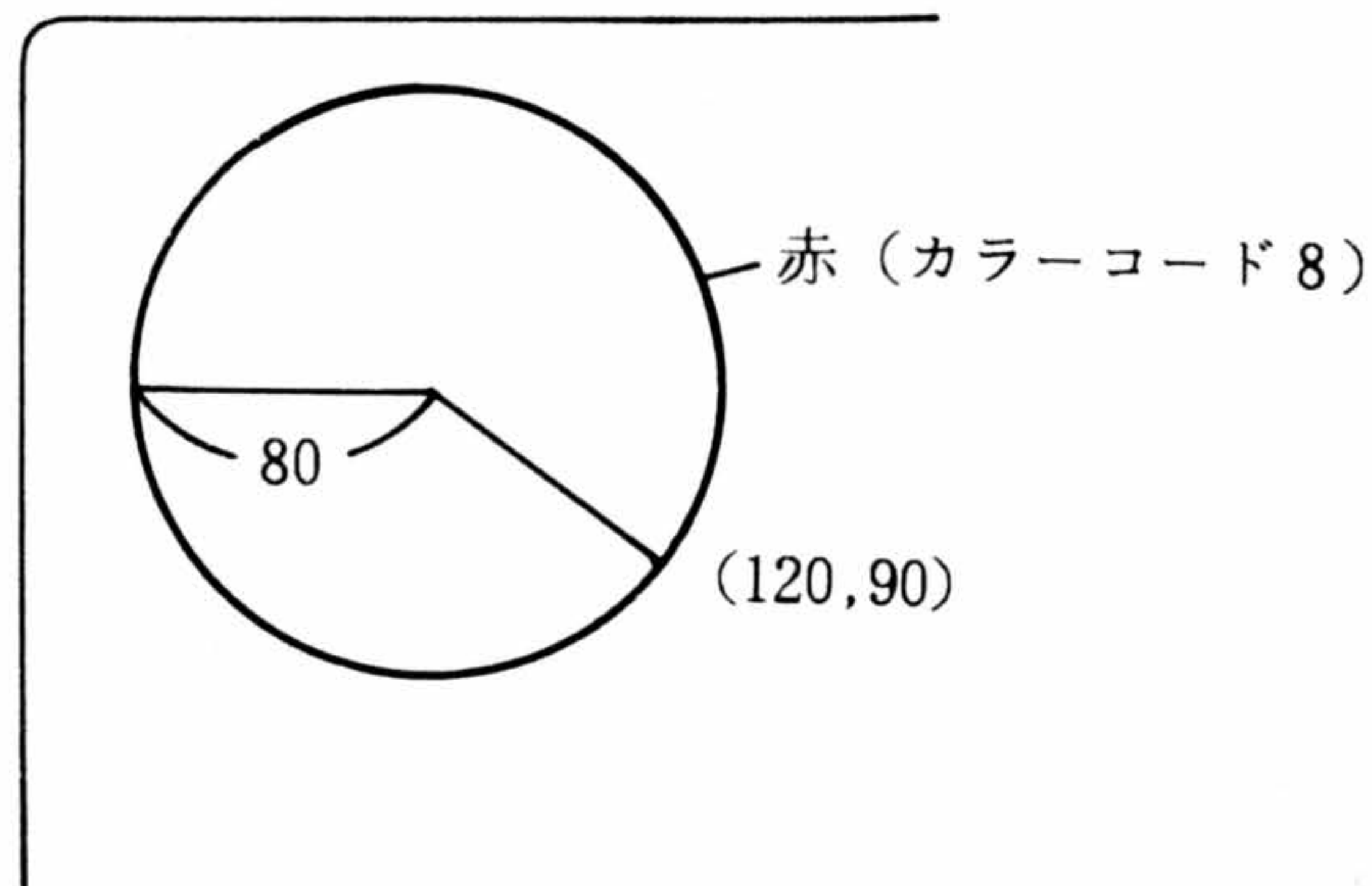
ＣＩＲＣＬＥ　(100，100)，50，，，，２

座標（100，100）を中心に、垂直方向の半径が50の縦長のだ円が表示される。

(ラジアン)

ラジアンは180°を１πとした単位でπは約 3.14である。角度をラジアンに直すときは次の式を使用する。

度数×3.14／１８０（ラジアン）

たとえば、90°は、90＊3.14／180 ＝1.57ラジアンとなる。

（例）
```
10 SCREEN2
20 CIRCLE (120 , 90), 80, 8
30 GOTO 30
```

これは、座標（120 , 90）を中心に半径80の赤い円を表示する。

注

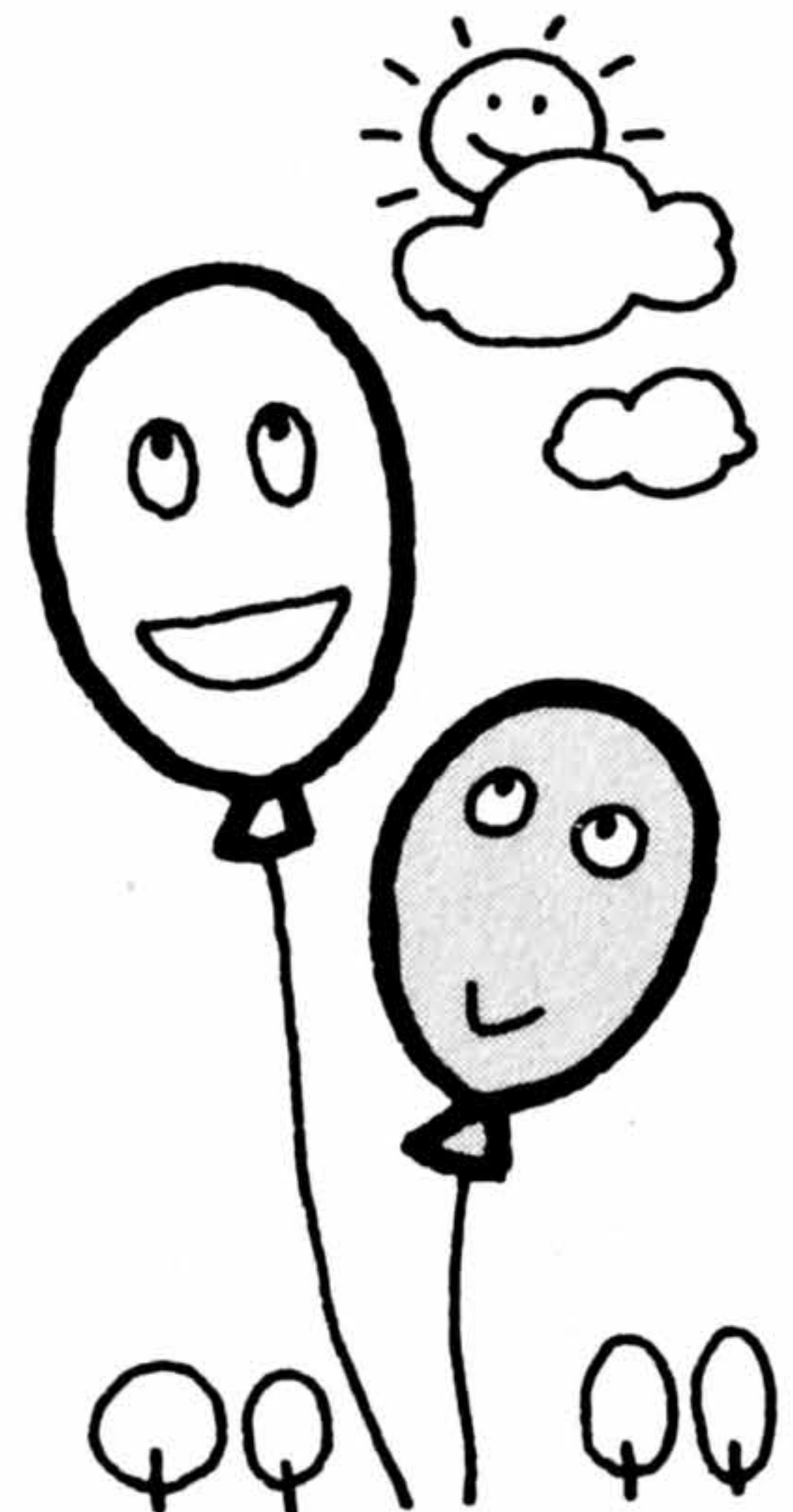
1　この命令はグラフィックモードのときだけ使用できる。

2　開始角度や終了角度は、－２π～２πの範囲を越えるとIllegal function callエラーになる。

3　ＣＩＲＣＬＥ文で指定する値は、変数や計算式でもかまわない。

4　（ｘ，ｙ）の前にＳＴＥＰをつけて、ＣＩＲＣＬＥ　ＳＴＥＰ（ｘ，ｙ）……とすれば、指定する円の中心位置は、一番最近表示された座標位置からの相対位置となる。

●サンプルプログラム

```
100 SCREEN2
110 FOR N=1TO100 STEP 10
120 CIRCLE(100,100),100-N,C
130 C=C+1
140 NEXT
150 C=2
160 FOR N=1TO100 STEP 10
170 CIRCLE(150,70),100-N,C
180 C=C+1
190 NEXT
200 C=2
210 FOR N=1TO100 STEP 10
220 CIRCLE(50,140),100-N,C
230 C=C+1
240 NEXT
250 C=2
260 FOR N=1TO100 STEP 10
270 CIRCLE(170,160),100-N,C
280 C=C+1
290 NEXT
300 GOTO 300
```

ＰＡＩＮＴ（ペイント）

■画面の一部を色で塗りつぶす

ＰＡＩＮＴ文は（ｘ，ｙ）で指定する座標を含む境界色で囲まれた領域を指定された領域色で塗りつぶす。

書式　ＰＡＩＮＴ　（ｘ，ｙ），領域色，境界色

説明　指定する座標位置（ｘ，ｙ）は、境界色で囲まれた領域内のどこでもいい。領域色、境界色はカラーコードで指定する。ただし、境界色を指定できるのは、マルチカラーモード（ＳＣＲＥＥＮ　３と５〜８）のときだけである。高解像度グラフィックモード（ＳＣＲＥＥＮ　２と４）のときは、カラーコードで指定した領域色と同じ色で囲まれた部分を塗りつぶす。

領域色の指定を省略すると、ＣＯＬＯＲ命令で指定した前景色が使われる。境界色を省略すると領域色が境界色として用いられる。

書式例　ＰＡＩＮＴ　（70，40），２，７

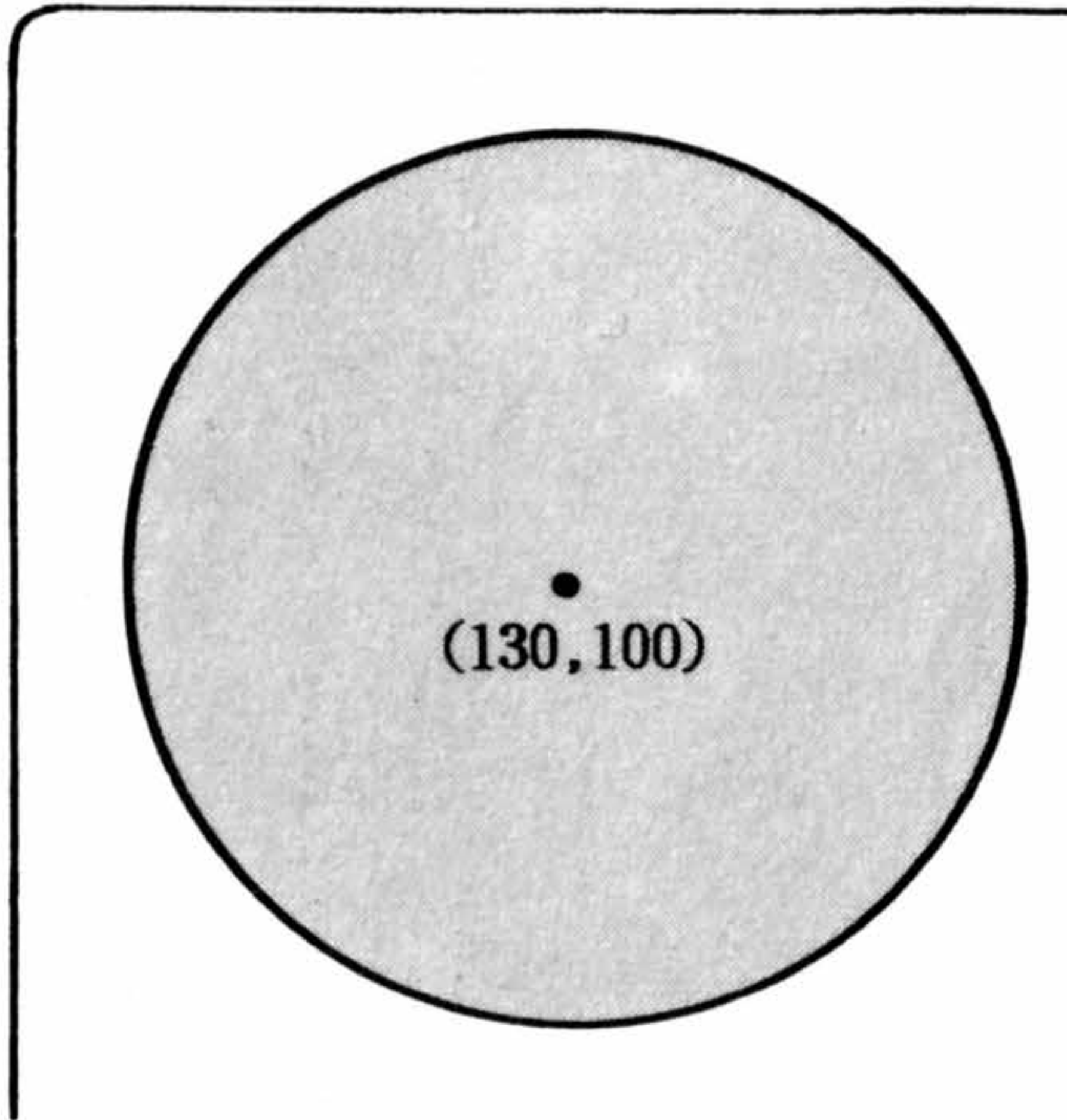
座標（70，40）の点を含み、水色(カラーコードの7)で囲まれた部分を、緑色（カラーコードの２）で塗りつぶす。

ＰＡＩＮＴ　（90，60），12

座標（90，60）の点を含み、紫色(カラーコードの12)で囲まれた部分を、同じ色（紫色）で塗りつぶす。

ＰＡＩＮＴ　（100　，80），，５

座標（100　，80）の点を含み、明るい青（カラーコードの５）で囲まれた部分を、ＣＯＬＯＲ命令で指定した前景色で塗りつぶす。

(例)
```
10 SCREEN 3
20 CIRCLE (130 , 100 ), 50, 10
30 PAINT (130 , 100 ), 8, 10
40 GOTO 40
```

(130,100)

上のプログラムは、20行で表示した黄色い円の内側を、赤で塗りつぶす。円の外側を塗りつぶすなら、30行のＰＡＩＮＴ文で指定した座標の値を、円の外側の座標値，たとえば（10，10）に変える。

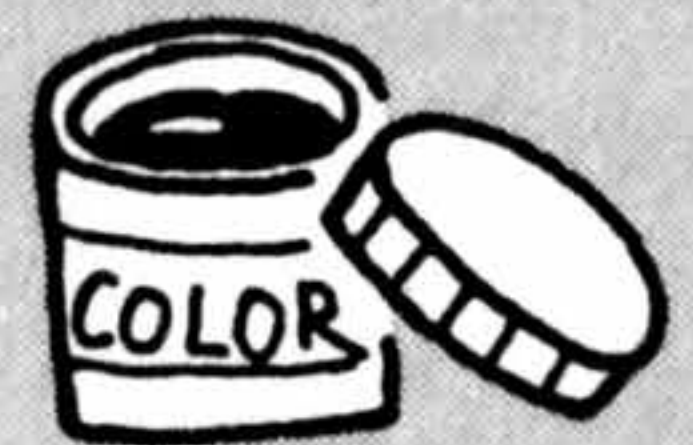
注

1　この命令は、グラフィックモードのときだけ使用できる。

2　（x，y）の前にＳＴＥＰをつけて、ＰＡＩＮＴ　ＳＴＥＰ（x，y）…とすれば、一番最近表示された座標位置からの相対位置となる。

●サンプルプログラム

```
100 SCREEN 2
110 CIRCLE(130,100),70,2
120 PAINT(130,100),2
130 CIRCLE(100,140),82,4
140 PAINT(100,140), 4
150 CIRCLE( 50,70),59,13
160 PAINT( 50, 70),13
170 CIRCLE(150,170),23,9
180 PAINT(150,170), 9
190 CIRCLE(140,25),40,5
200 PAINT(140,25), 5
210 GOTO210
```

COLOR

POINT（ポイント）

■指定した位置の色を調べる

グラフィック画面で、任意の座標位置のドットの色を調べ、その色のカラーコードを得る。

書式 POINT （x, y）

説明 座標（x，y）と位置を指定して、そこに表示されているドットの色を調べ、その結果をカラーコードで得る。ただし、指定された座標にスプライトパターンが表示されていても、スプライトパターンの色を調べることはできない。

またテキストモードでは使用できないので、テキストモードのキャラクタの色は判断できないが、グラフィック画面に表示したキャラクタの色は判断する。

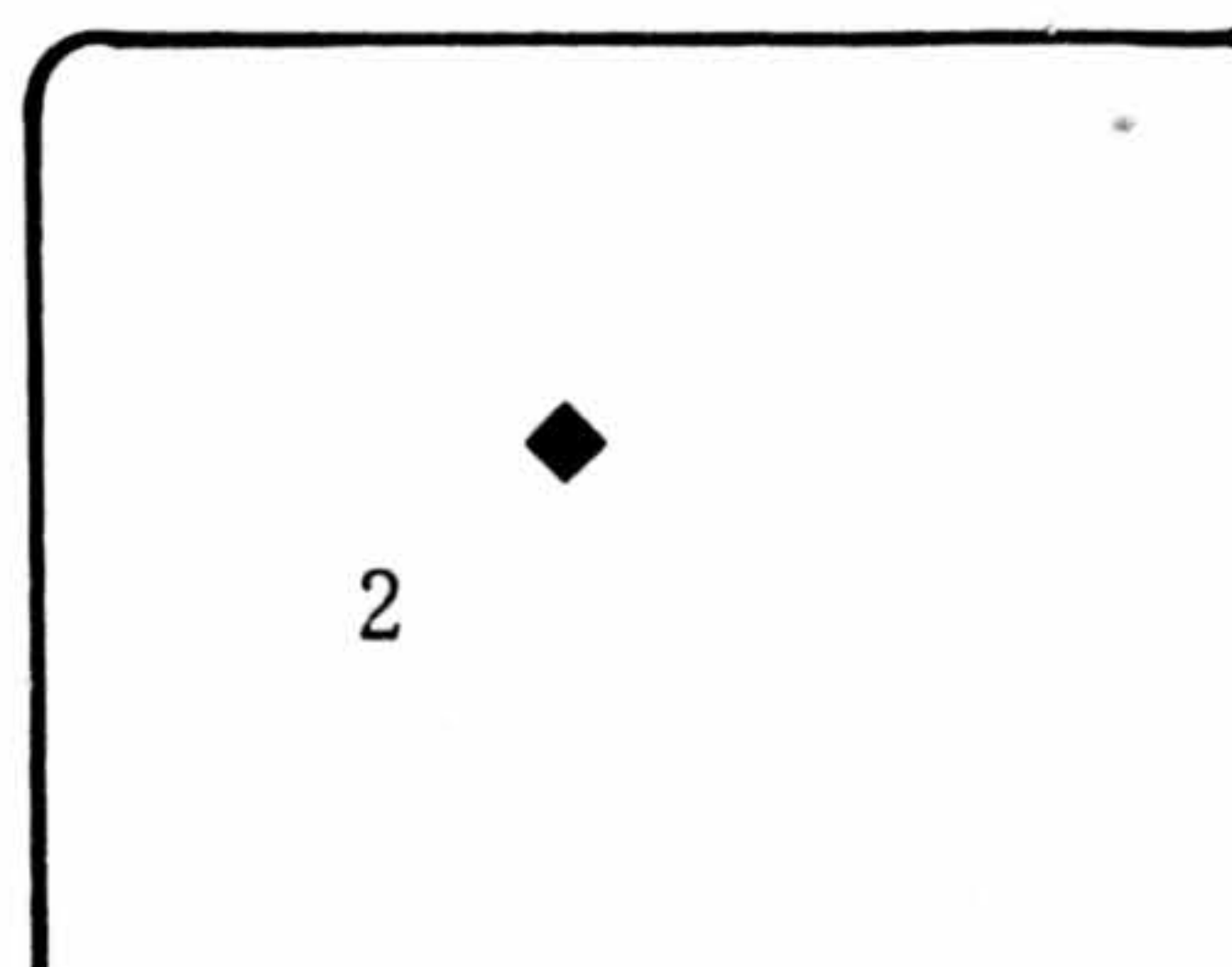
書式例 POINT （100 , 100）

座標（100 , 100）に表示されているドットの色を調べその色のカラーコードを得る。

（例）

```
10 SCREEN 2
20 OPEN "GRP:" FOR OUTPUT AS #1
30 PRESET (130 , 100 )
40 COLOR 2
50 PRINT #1, "◆"
60 A=POINT (134 , 104)
70 PRESET (10 , 120)
80 PRINT #1, A
90 GOTO 90
```

◆

2

このプログラムは、グラフィック画面に表示されたダイヤのマークの色を、60行のPOINT文で調べ、その色のカラーコードを画面に表示する。

（グラフィック画面のキャラクタ表示）

上のプログラムでは、グラフィック画面にキャラクタを表示させている。

まず20行のＯＰＥＮ文だが、これはグラフィック画面にキャラクタを表示させるときのきまり文句として、プログラムの最初に実行しておく。

つぎに30行で、キャラクタを表示させる場所を決める。グラフィック画面に表示するときは、ＬＯＣＡＴＥは使えない。

40行で、表示する色を決めている。１文字ごとに自由に色をつけることもできる。

50行は、実際に画面に表示させる命令である。テキストモードのＰＲＩＮＴ文と書き方は少し違うが、使い方は同じである。

このようにすれば、グラフィック画面にキャラクタが表示できる。

●サンプルプログラム

```
100 SCREEN 2
110 OPEN"grp:"FOR OUTPUT AS #1
120 CIRCLE(130,100),70,2
130 PAINT(130,100),2
140 A=POINT(130,99)
150 PRINT#1,A
160 CIRCLE(100,140),82,4
170 PAINT(100,140), 4
180 A=POINT(100,140)
190 PRINT#1,A
200 CIRCLE( 50,70),59,13
210 PAINT( 50, 70),13
220 A=POINT( 50, 70)
230 PRINT#1,A
240 CIRCLE(150,170),23,9
250 PAINT(150,170), 9
260 A=POINT(150,170)
270 PRINT#1,A
280 CIRCLE(140,25),40,5
290 PAINT(140,25), 5
300 A=POINT(140,25)
310 PRINT#1,A
320 GOTO320
```

DRAW（ドロウ）

■画面に点を動かして図形を描く

グラフィック画面上にある１点（参照点）を相対的に移動させながら図形を描いていく。

書式　DROW　"文字式"

説明　文字式は参照点を移動させるグラフィックマクロ命令である。文字式は文字列でも文字変数でもかまわない。文字式には、位置の設定、移動の方向と距離、角度などを指定するグラフィックマクロ命令を１つ以上連続して書く。

書式例　DRAW　"D10R20U10L20"

" "で囲まれたグラフィックマクロ命令に従って図形を描く。

DRAW　A$

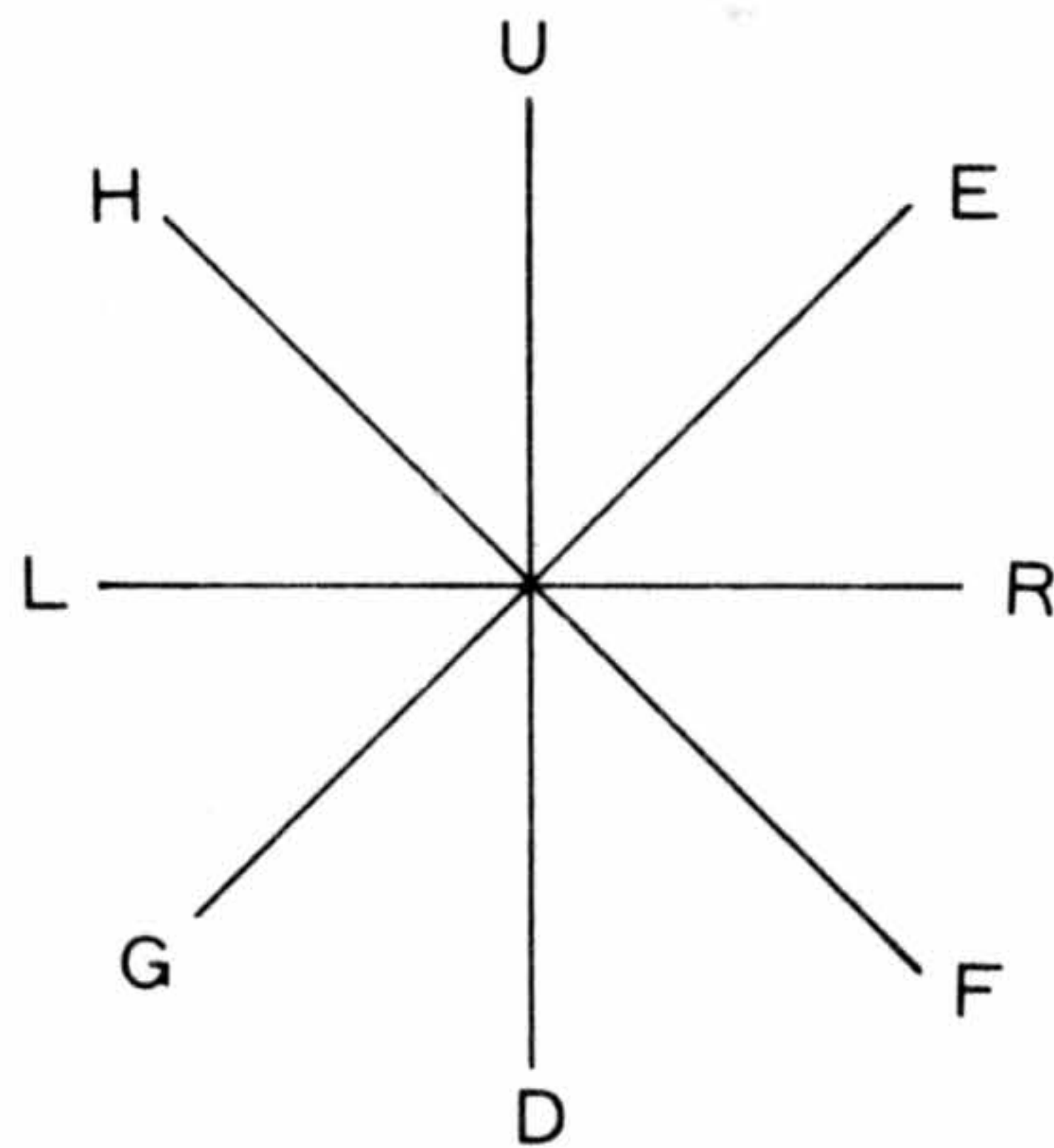
A$に代入されたグラフィックマクロ命令に従って図形を描く。

●グラフィックマクロ命令

１．移動マクロ命令

Un………　上にnだけ移動する。
Dn………　下にnだけ移動する。
Ln………　左にnだけ移動する。
Rn………　右にnだけ移動する。
En………　右ななめ上にnだけ移動する。
Fn………　右ななめ下にnだけ移動する。
Gn………　左ななめ下にnだけ移動する。
Hn………　左ななめ上にnだけ移動する。

以上の各コマンドは、最後に参照された位置から移動させるものである。移動距離は、n×倍率（Sコマンドで指定）となる。このときのnはドット数（要素単位）で指定される。

Mx，y……　座標（x，y）に移動する。xに＋または－の符号がついて

いれば、相対移動である。たとえば、M＋４，３なら現在の参照点よりｘ軸方向に＋４、ｙ軸方向に＋３の移動である。

Ｂ……… 上記の命令で各方向に移動するが、線は描かない。

Ｎ……… 上記の命令で点を表示しながら移動するが、表示後の参照点（点の表示を始める位置）は始点に戻る。

ＢコマンドとＮコマンドはともに、上記の移動コマンドの前に記述して使用する。

（例） ＤＲＡＷ "ＮＬ10Ｄ４"

２．回転マクロ命令

Ａｎ…… 移動マクロ命令で表示する図形を90°単位で回転して表示する。

ｎは角度で０～３の数で設定する。0は 0°、1は90°、2は180°、3は270°である。

３．色指定マクロ命令

Ｃｎ…… 図形を表示する色を指定する。ｎは０～15のカラーコードである。

４．スケールファクタマクロ命令

Ｓｎ…… 移動マクロ命令などで指定した移動距離と実際に移動する距離との比率（スケールファクタ）を指定する。

ｎは 1～255 の数で指定し、ｎの４分の１が比率となる。たとえば、Ｓ８とすれば、スケールファクタは８÷４で２ということになる。つまり、移動マクロ命令によって移動する距離は、指定した距離にこのスクールファクタをかけたものである。

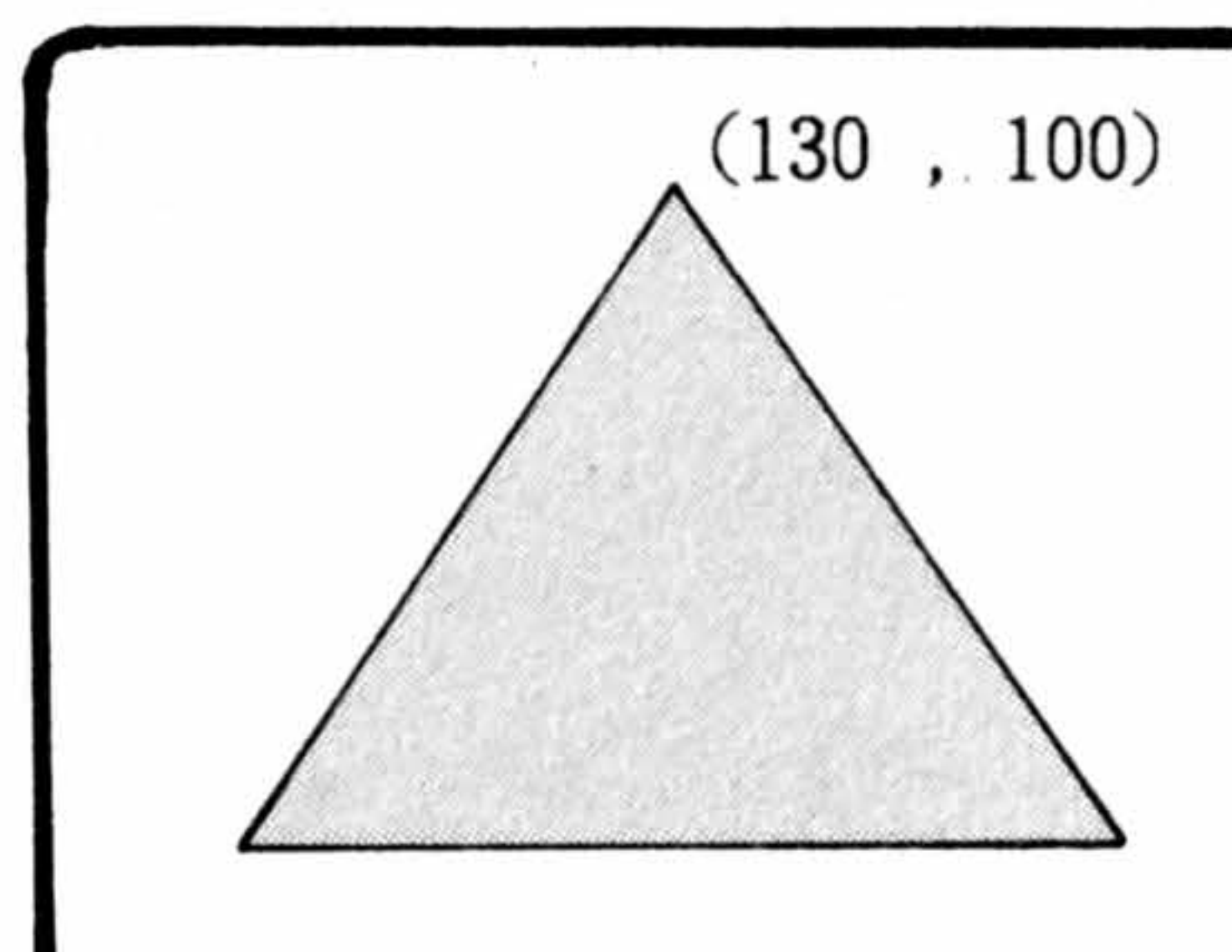
スクールファクタの指定を省略すれば、４が設定されるので、比率は１となり、移動マクロ命令で指定した距離と実際の移動距離が同じになる。

５．ローカルマクロ命令

Ｘ文字変数… 文字変数の中に定義されているグラフィックマクロ命令を実行する。この場合は、文字変数のあとに必ずセミコロンをつける。

（例） Ａ＄＝"Ｕ20Ｌ20Ｄ20"：ＤＲＡＷ"Ａ＄；"

（例）　10　SCREEN　2
　　　　20　PSET　(130 , 100), 2
　　　　30　DRAW　"G40R80H40"
　　　　40　GOTO　40

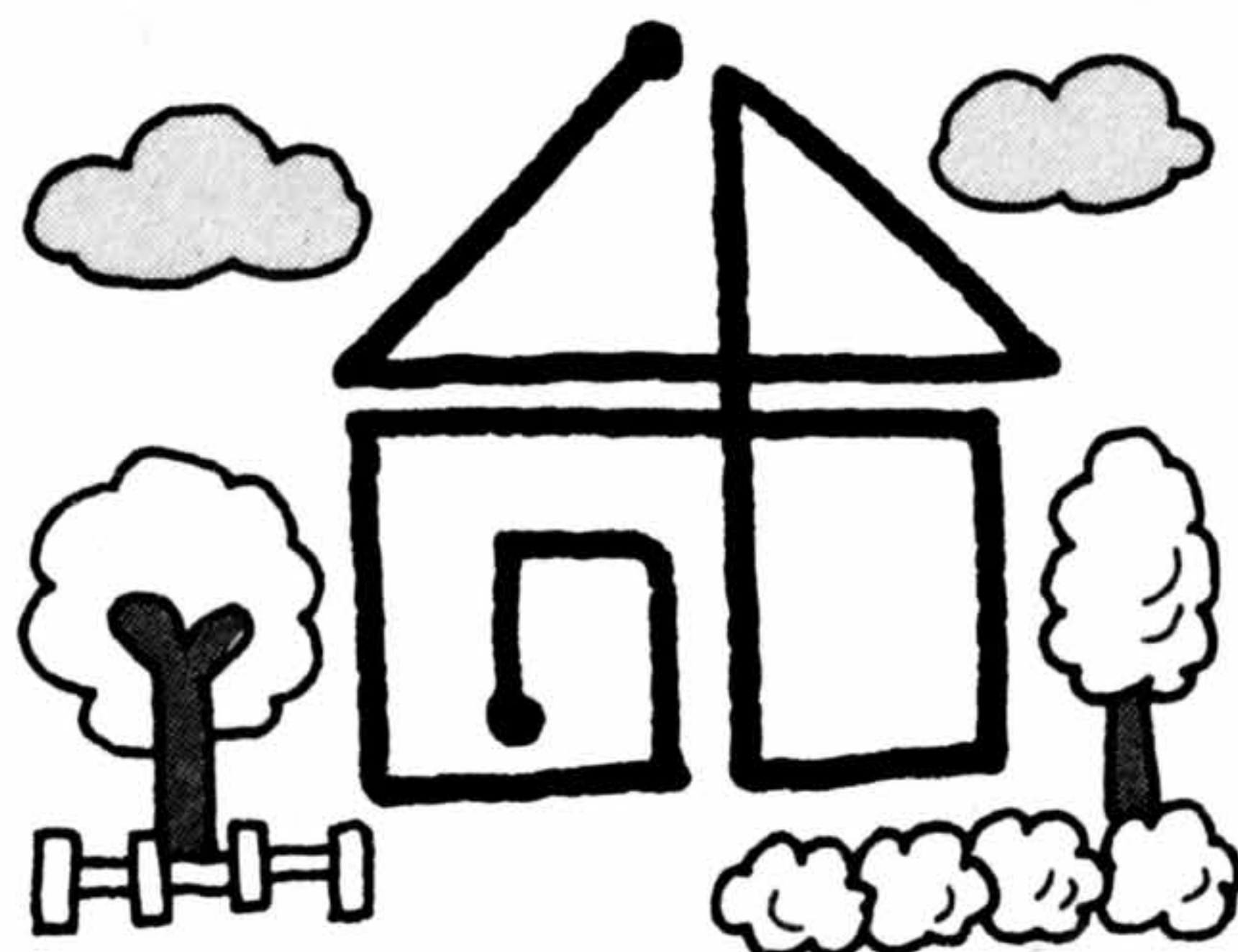
(130 , 100)

画面に緑色の三角形を表示するプログラムである。

ＤＲＡＷは画面に図形を描くのにとても便利な命令なのである。例えば25行に、ＤＲＡＷ　"Ａ１"を加えるだけで三角形を横に向かせることができる。

注

1　この命令は、グラフィックモードのときだけ使用できる。

●サンプルプログラム

```
100 SCREEN 2
110 PSET(100,100),4
120 DRAW"A0"
130 DRAW "G40r80h40"
140 FOR N=1TO500:NEXT
150 PSET(100,100),7
160 DRAW"A1"
170 DRAW "G40r80h40"
180 FOR N=1TO500:NEXT
190 PSET(100,100),3
200 DRAW"A2"
210 DRAW "G40r80h40"
220 FOR N=1TO500:NEXT
230 PSET(100,100),9
240 DRAW"A3"
250 DRAW "G40r80h40"
260 GOTO260
```

COPY（コピー）

■グラフィック画面を複写する

書式

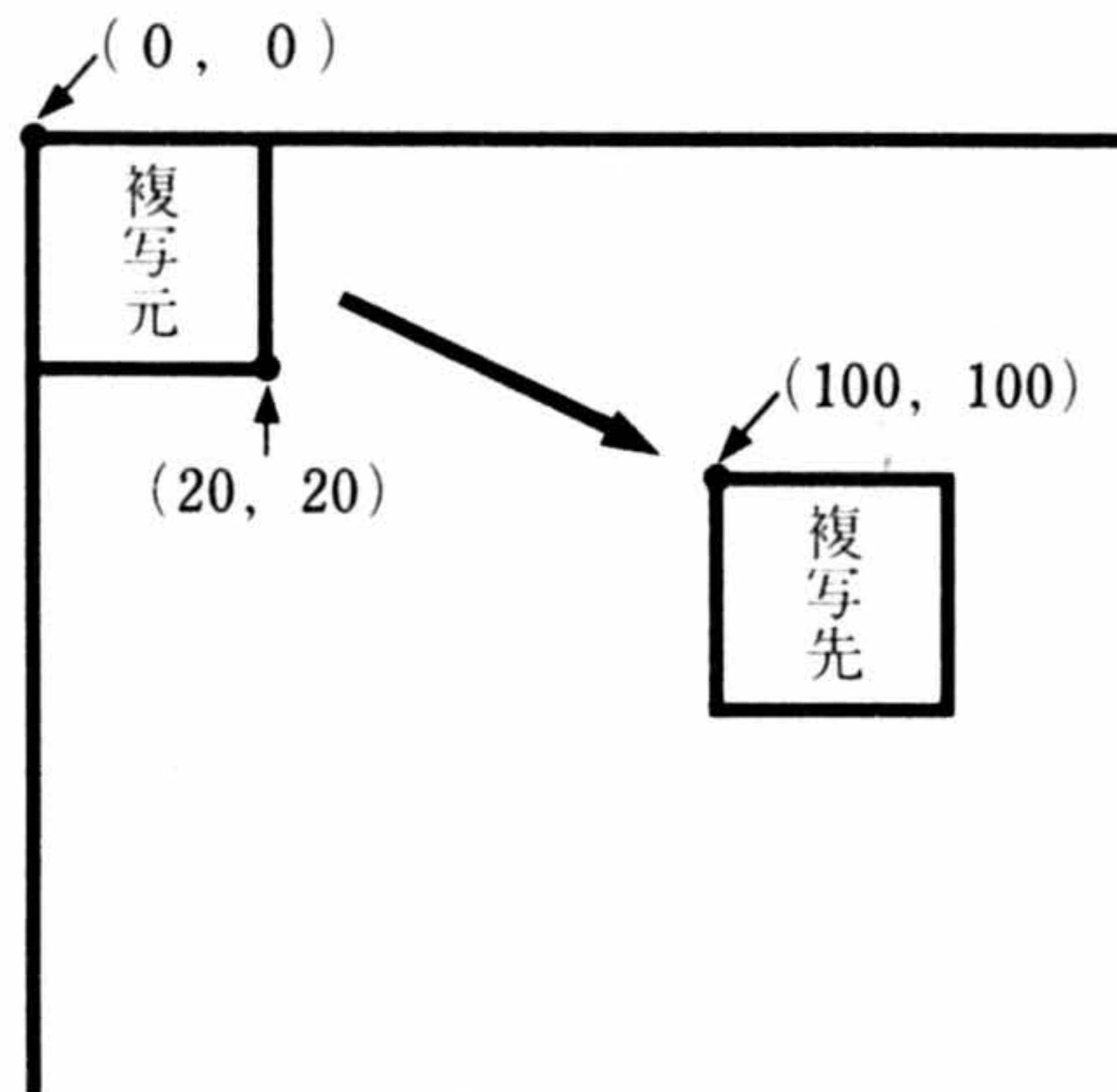
COPY （X, Y）－（X1, Y1）[, 〈複写元のページ〉]
TO （X2, Y2）[, 〈複写先のページ〉]

説明　画面領域内の複写およびディスクファイル、配列にコピーする。画面上のグラフィック画面を他の場合へ複写するときに使い、（X, Y）－（X1, Y1）でグラフィック領域の大きさを指定し、（X2, Y2）座標に複写する。

複写ページが同じ場合には、ページ指定は省略できる。

書式例　COPY （0, 0）－（20, 20）, 0　TO （100, 100）

左上座標（0, 0）から右下座標（20, 20）の範囲のグラフィック図形を、複写先の左上座標（100, 100）に複写する。

（0, 0）
複写元
（20, 20）
（100, 100）
複写先

ディスクファイルおよび配列にコピーするには、次の書式を用いる。

書式　COPY （X, Y）－（X1, Y1）[, 〈複写元のページ〉]

TO ［〈ファイル名〉
　　〈配列名〉］

説明　長方形の領域を指定してディスクファイルおよび配列に記憶させておくことができる。ただし、あらかじめDIM文で、必要なバイト数以上を宣言して、領域を確保しておかなければならない。

PUT KANJI（プットカンジ）

■MSX 2の画面に漢字を表示させる

書式 PUT KANJI [X, Y],（漢字コード）[,（色）]

説明 漢字をグラフィック画面に表示させる命令で、SCREEN 5～8のビットマップグラフィックモードに限り有効である。[X, Y] は、漢字を表示させる画面の左上の座標位置を指定し、次に表示したい文字のJISコードを指定する。

色はカラーコードで指定し、省略した場合には、COLOR文で指定した前景色になる。

なお、漢字を表示させるには、漢字ROMが必要である。

書式例 PUT KANJI（0, 0), &h3441

左上座標（0, 0）に「漢」という字を表示する。ここでは座標を指定しているが、座標値は省略することもできる。省略した場合、文字は基準位置を左上とするように表示される。

（例）
```
 10 SCREEN 5
 20 CLS
 30 PRESET (8, 16)
 40 READ A
 50 IF A=-1 THEN 80
 60 PUT KANJI, A
 70 GOTO 40
 80 Z$=INPUT$ (1)
 90 END
100 DATA &h234D, &h2353, &h2358, -1
```

左上座標（8, 16）の場所から「MSX」と表示される。40行でDATA文によって漢字コードを読み込み、60行で漢字を表示させている。

⑤音楽・音・ブザー

PLAY（プレイ）

■音楽を演奏する

ミュージックマクロ命令を使用して、MSX・コンピュータが音楽を演奏する。単音階の演奏から二重和音，三重和音まで演奏することができる。

書式 PLAY "文字式1"，"文字式2"，"文字式3"

説明 文字式はすべてミュージックマクロ命令で表現された文字列である。ミュージックマクロ命令には、音の高さ、音の長さ、音の速度（テンポ）、音の大きさ、音色、ローカルマクロの6種類がある。

各文字列をコンマ（,）で区切り、最大3重和音まで表現できる。文字式1，文字式2，文字式3はそれぞれボイスチャンネルの1，2，3に相当する。音を出さないボイスチャンネルの文字式は省略する。

文字式は文字変数で指定してもかまわない。

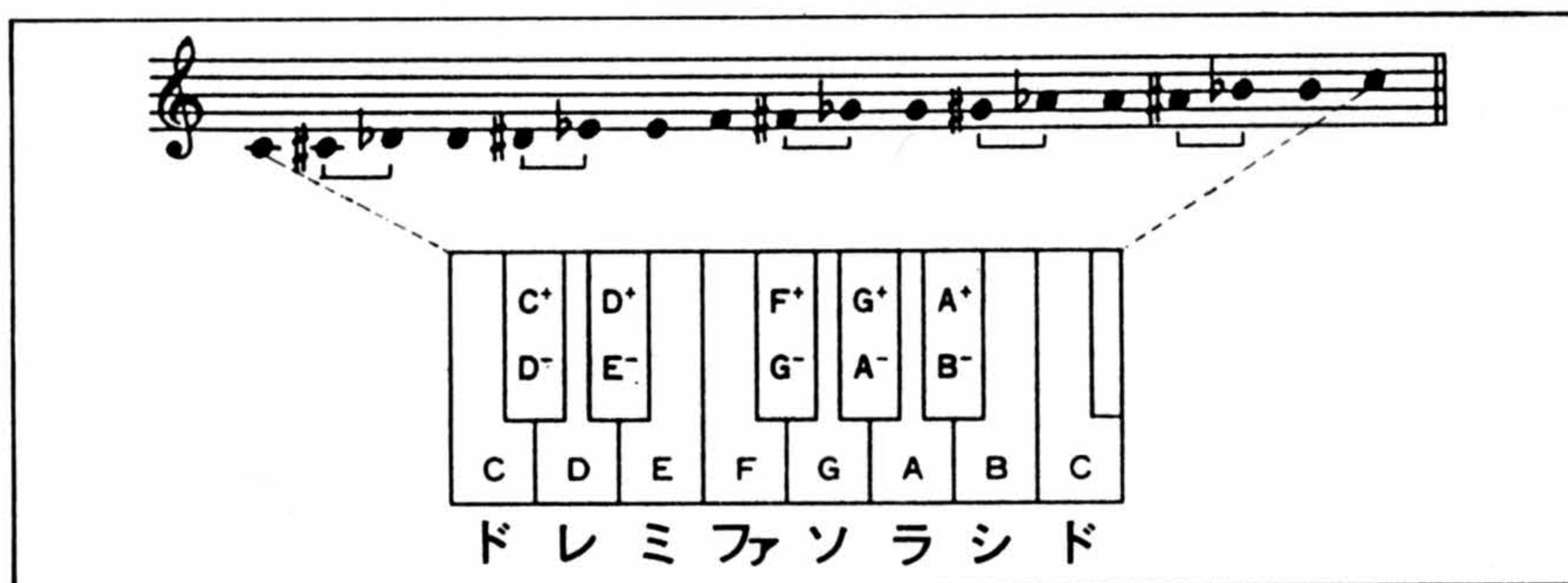
書式例 PLAY "CDEFGABO5C"
PLAY A$+B$，C$

●ミュージックマクロ命令

（音の高さ）

アルファベットのABCDEFGで指定する。ハ長調ならドはC，レはD，ミはE，以下F，G，A，Bの順である。

ド レ ミ ファ ソ ラ シ （ハ長調）
C D E F G A B

（例） PLAY "CEG"……ドミソ

半音上げるときはアルファベットのあとに#または＋を書く。

（例）　C#またはC＋………ドの半音上がり

半音下げるときはアルファベットのあとに－（マイナス）を書く。

（例）　C－………ドの半音下がり

（オクターブ）

音域は1～8の数値で8オクターブ指定できる。

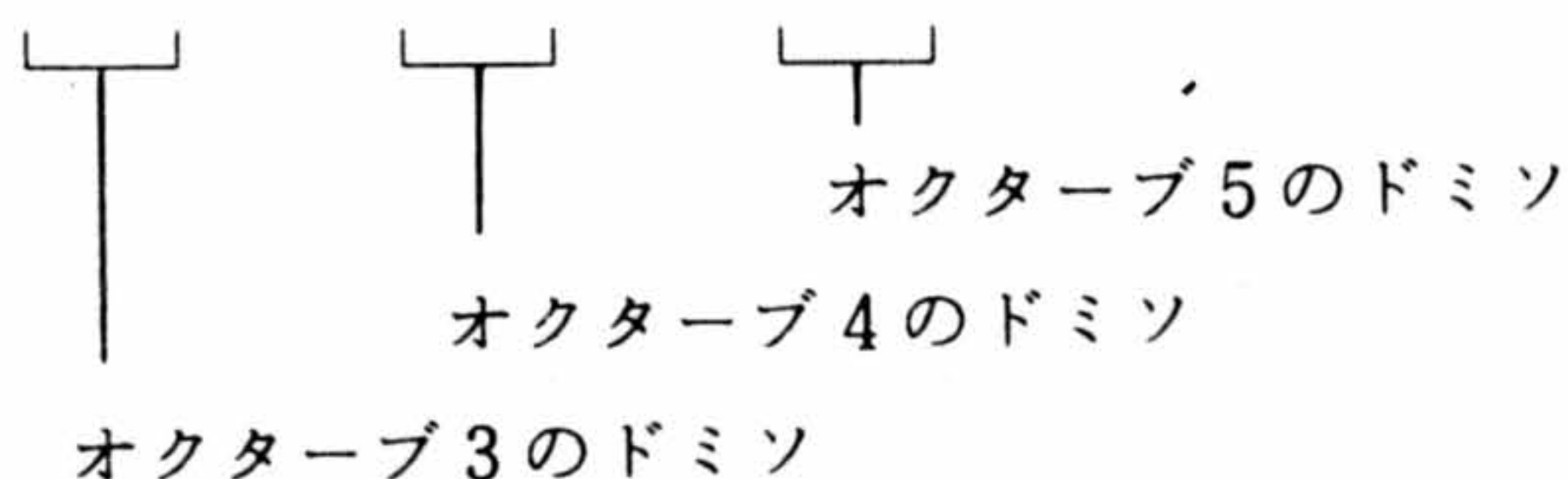
書式　On

Oはオクターブ指定の命令。nは1～8の数値。1は最低音音域，8は最高音音域である。一度オクターブを指定すると、つぎのオクターブ指定まで有効である。初期設定値はO4で、1点ハ音から始まるオクターブを示す。

（例）　PLAY　"O3CEGO4CEGO5CEG"

オクターブ5のドミソ

オクターブ4のドミソ

オクターブ3のドミソ

文字式のはじめになにも指定しなければO4である。

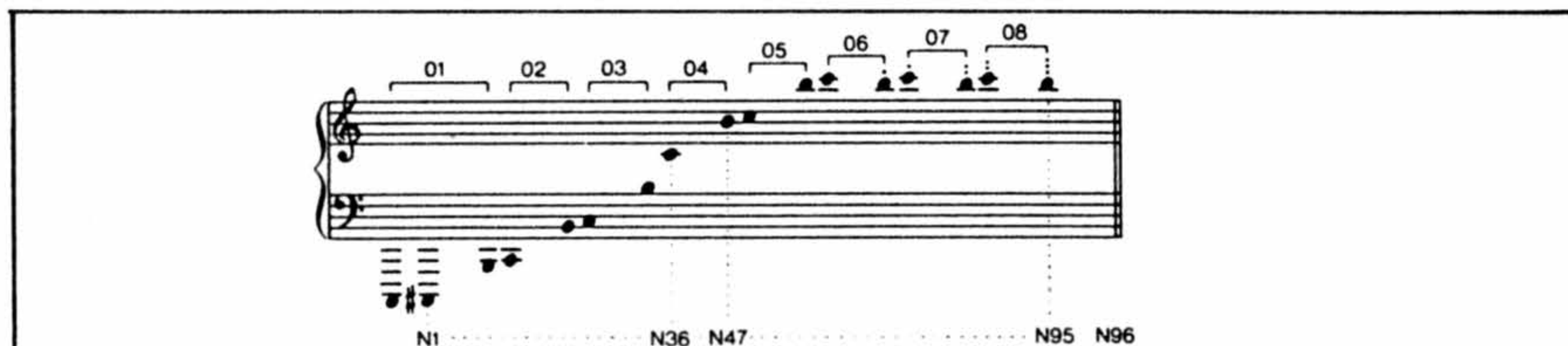
O1からO8のオクターブを通して、各音の音の高さを数値で表現することもできる。

書式　Nn

nは0～96の数値である。O1のC#をN1として、O8のBに相当するN95まで、半音上がるごとにnが1ずつふえる。

ただし、Ｎ０はＲ（休符）と同じ働きをする。

（例）　ＰＬＡＹ　"Ｎ36Ｎ37Ｎ38Ｎ39Ｎ40Ｎ41Ｎ42Ｎ43"

（音の高さＯとＮの対応表）

音と高さＯとＮの対応表

	Ｏ１		Ｏ２		Ｏ３		Ｏ４		Ｏ５		Ｏ６		Ｏ７		Ｏ８	
Ｃ　ド		1	12	13	24	25	36	37	48	49	60	61	72	73	84	85
Ｄ　レ	2	3	14	15	26	27	38	39	50	51	62	63	74	75	86	87
Ｅ　ミ	4		16		28		40		52		64		76		88	
Ｆ　ファ	5	6	17	18	29	30	41	42	53	54	65	66	77	78	89	90
Ｇ　ソ	7	8	19	20	31	32	43	44	55	56	67	68	79	80	91	92
Ａ　ラ	9	10	21	22	33	34	45	46	57	58	69	70	81	82	93	94
Ｂ　シ	11		23		35		47		59		71		83		95	

上の表では、１オクターブごとにＮの数字はそれぞれ２列ある。左はその音の正音、右がその音の半音上の音である。たとえばＯ４Ｃ＃のときのｎの値は37ということになり、Ｎ37がＯ４Ｃ＃と同じということになる。

（音の長さ）

音の長さは 1〜64の数値で指定する。

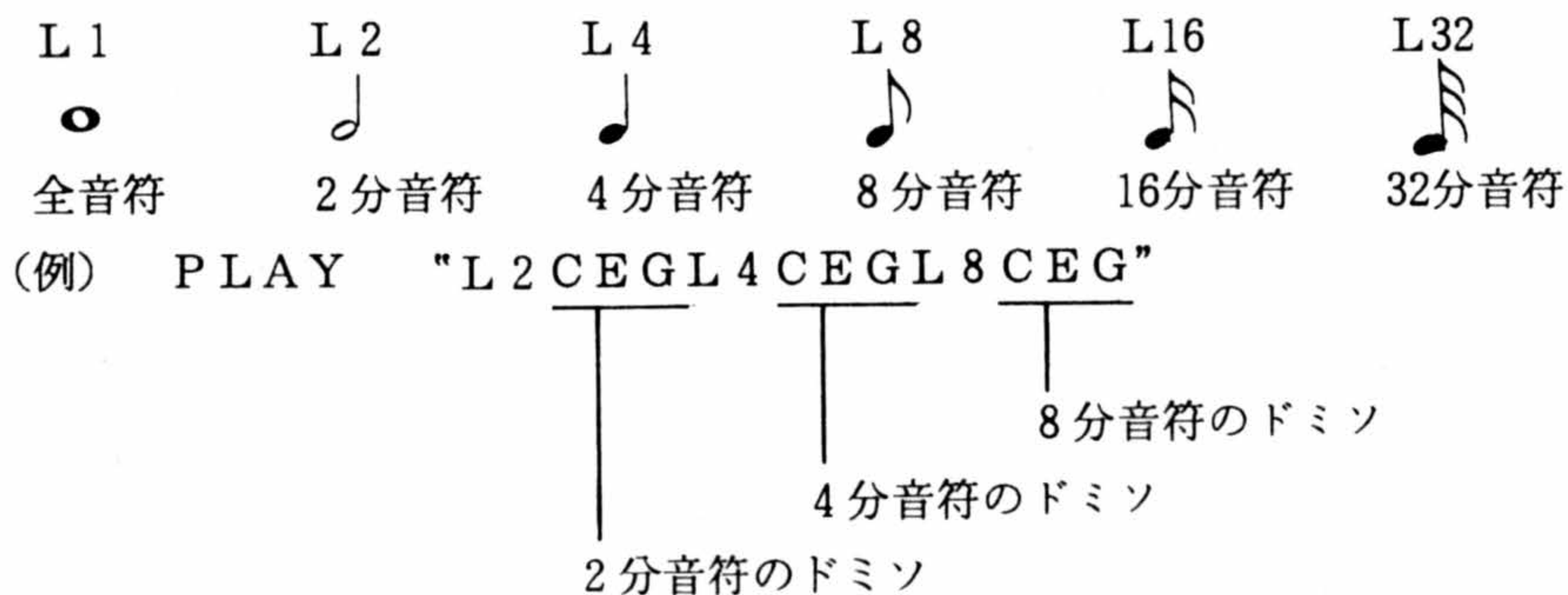
書式　Ｌｎ

Ｌは長さを指定する命令、ｎは 1〜64の数値である。１を全音符（４拍）とし、２は２分音符（２拍）、４は４分音符（１拍）である。あらためて指定しなければ初期値はＬ４である。１度音の長さを指定すると、つぎに音の長さを指定するまで有効である。

また、１音の長さだけを変えたいときは、音階名（A～G）の後ろに音の長さを示す数値を付け加える。たとえば、Cの音だけの長さを変えるときは、C８というようにする。

L１	L２	L４	L８	L16	L32
𝅝	𝅗𝅥	♩	♪	𝅘𝅥𝅯	𝅘𝅥𝅰
全音符	２分音符	４分音符	８分音符	16分音符	32分音符

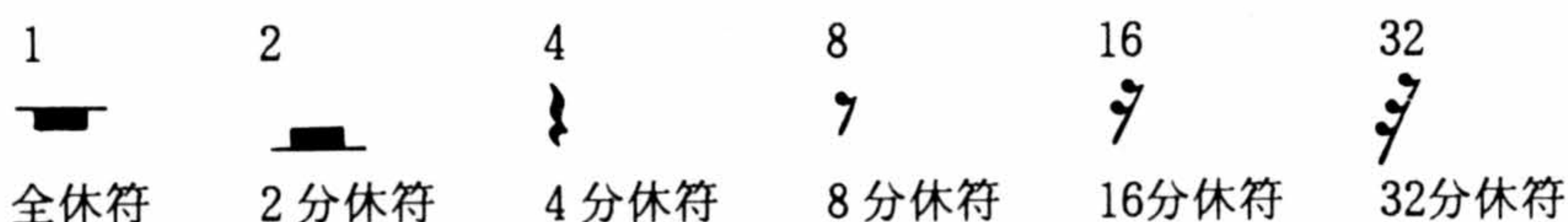
（例） PLAY "L２CEGL４CEGL８CEG"

- CEG（L２の後）：２分音符のドミソ
- CEG（L４の後）：４分音符のドミソ
- CEG（L８の後）：８分音符のドミソ

（休符）

音符のときの音の長さと同じように、音を休む長さ（休符）も１～64の数値で指定する。

書式 Rn

Rは休符の命令、nは休みの長さで、１～64の数値で示す。１は全休符、２は２分休符である。Rだけで数値を指定しなければ４分休符である。

1	2	4	8	16	32
𝄻	𝄼	𝄽	𝄾	𝄿	𝅀
全休符	２分休符	４分休符	８分休符	16分休符	32分休符

（付点音符・付点休符）

音階名（A～G）または休符（Rn）の後ろにピリオドを付けると、付点音符または付点休符となり、その音の長さまたは休みの長さが1.5 倍になる。また、２個並べて書くと９／４（2.25）倍３個並べて書けば27/8（3.375 ）倍になる。

（例） PALY "S９M3000T200 L４F. FF. GA. AA. GF. FF. DC２"

（テンポ）

音楽のテンポを指定する。

書式 Tn

Tはテンポの命令、nは1分間に演奏する4分音符の回数。32〜255 の数で指定する。なにも指定しなければT120 が設定される。

(例) PLAY "T120 O5L8"

(音の大きさ)

音量を指定する。

書式 Vn

Vは音の大きさを指定する命令、nは 0〜15の数値である。nの値が大きくなるほど音も大きくなる。nを指定しなければV8が設定される。一度音量を指定すると、つぎに音量を指定するまで有効である。

```
(例) 10 PLAY "V9T120 O5L8"
     20 PLAY "GF+GD4. O4B4. AG+AO5E4. E4."
     30 PLAY "DF+EEDCCO4B4. B4. B4."
```

(音色)

音色を決定するのは主として、エンベロープ周期を指定するMマクロ命令とエンベロープ形状を指定するSマクロ命令である。Sマクロ命令とVマクロ命令を同時に指定することはできない。

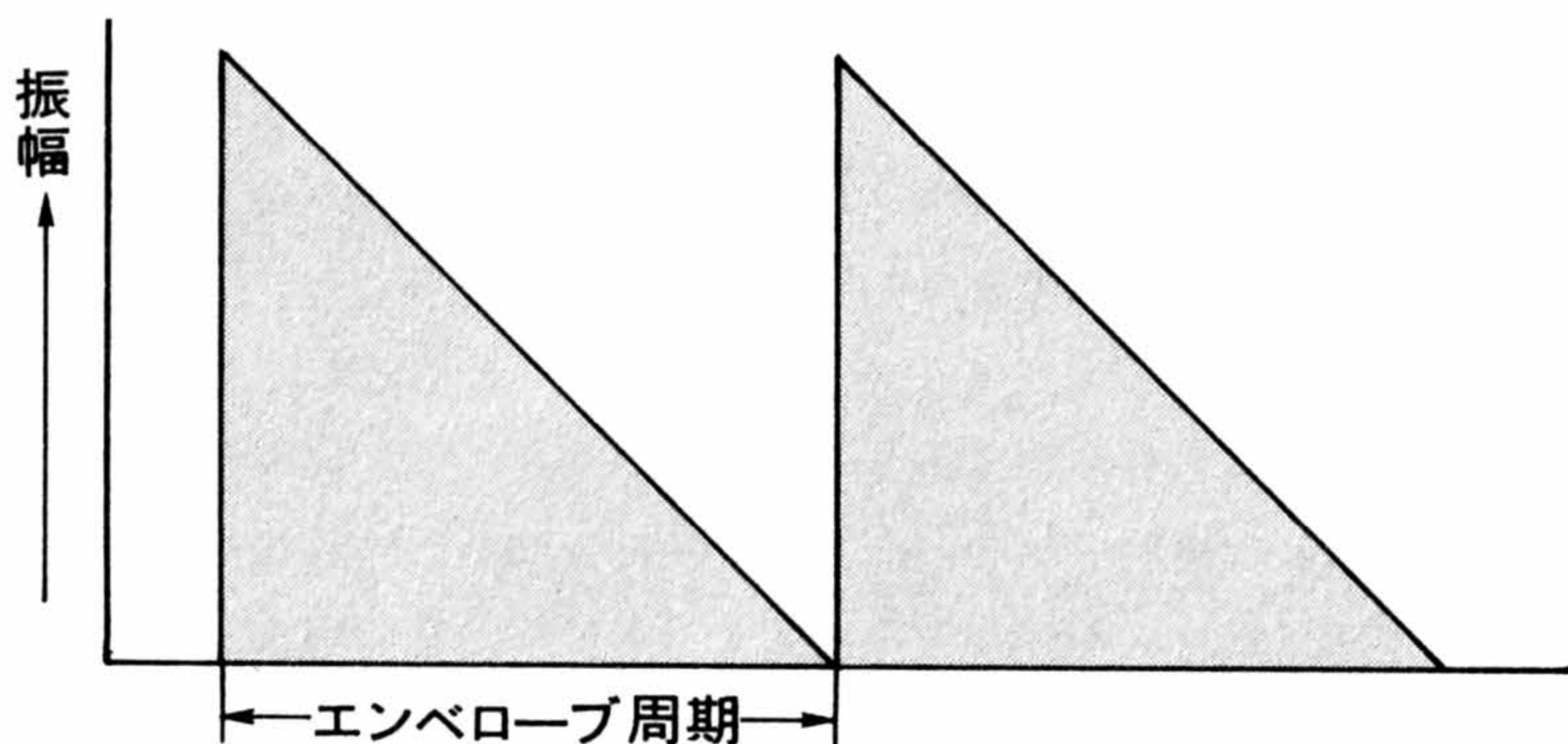
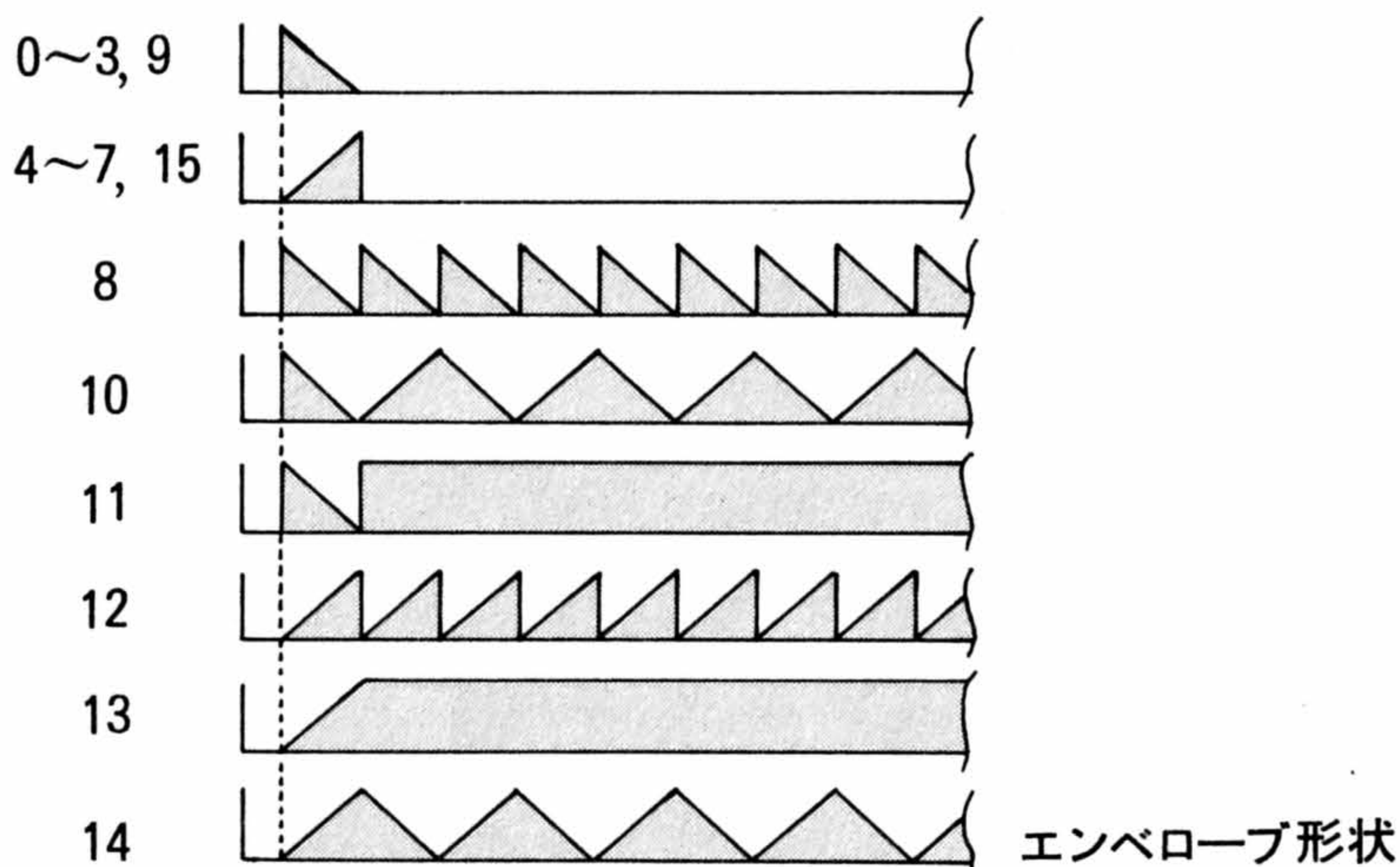
エンベロープ周期とエンベロープ形状はコンピュータ的な音に微妙な変化を与えるが、あまりにも専門的であり、実用するためには、Mマクロ、Vマクロ命令の数値の微妙な違いをいろいろ試して知っておかなければならない。

MSX・ミュージックの音の深みに関するもので、音の演奏そのものに絶対に必要だというものではないから、特に指定しなくても支障はない。

Sマクロ命令はエンベロープ形状(音量変化の波形)を指定する。

書式 Sn

nは 0〜15の数値である。

Sの値

この命令はVコマンドで音量を指定すれば解除される。

Mマクロ命令は音のエンベロープ周期を指定する。

書式 Mn

nは0〜65535の数値である。数値があまり大きいとエンベロープの特徴が出ない。

MコマンドとSコマンドの組合せで微妙な音色の変化を試みてみる。

（例） PLAY "V9CEG"

PLAY "S 9M2000CEG"
PLAY "S 8M 700CEG"
PLAY "S10M 500CEG"
PLAY "S11M3000CEG"
PLAY "S12M2000CEG"
PLAY "S13M 500CEG"
PLAY "S14M 800CEG"

(ローカルマクロ)

書式 X文字変数;

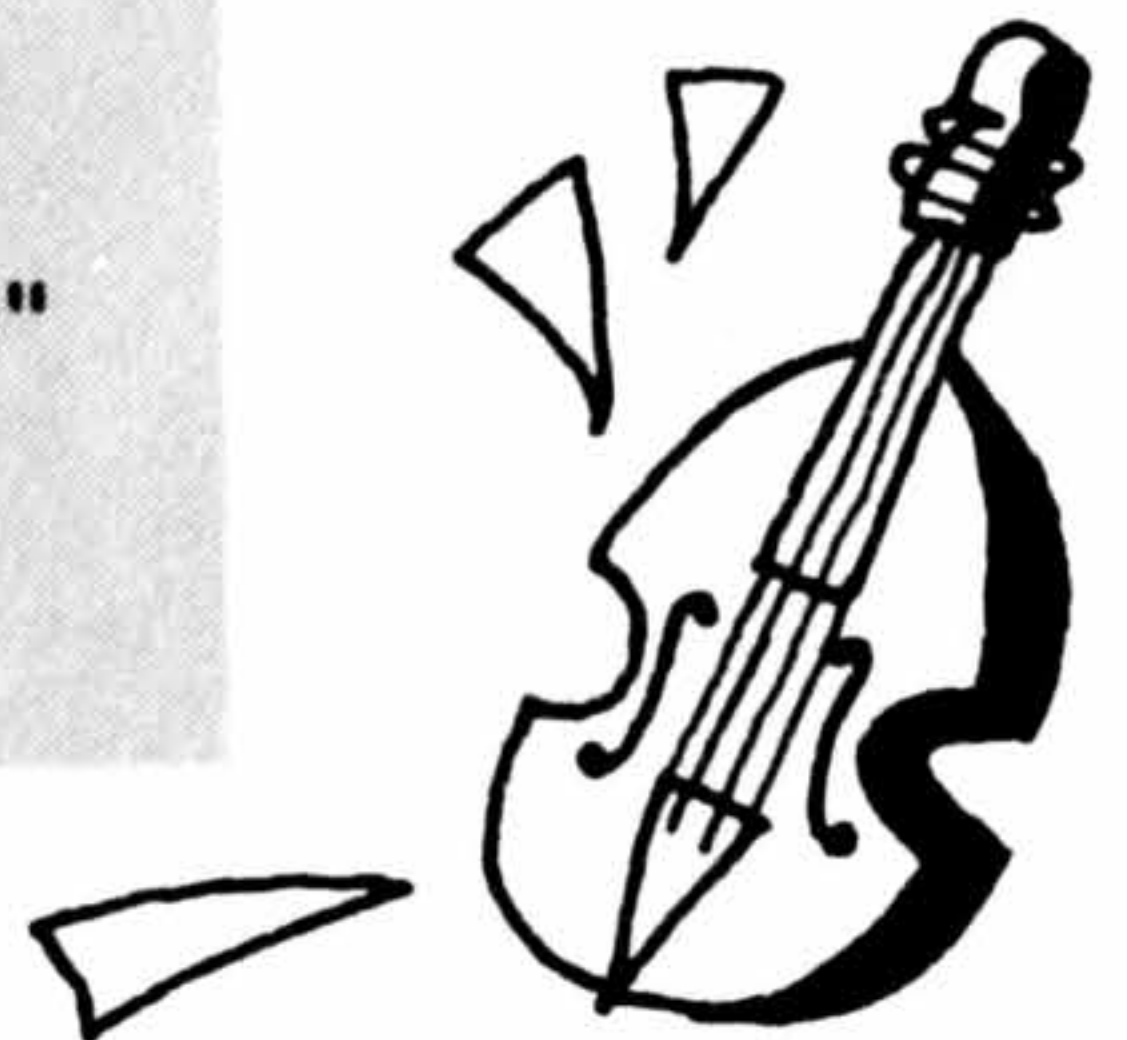
文字変数の中に含まれているミュージックマクロ命令を実行する。この命令はミュージックマクロ命令の文字列が255を越えて1行で書ききれない場合や、音を少しずつ変化させながら繰り返し演奏するときなどに使用する。

たとえば、PLAY "CDEFGAB" は、A$= "CDEFGAB:PLAY" XA$;と同じである。

文字変数のあとのセミコロンは、必ず入れる。

音楽演奏中にBEEP命令やSOUND命令で音を出すと、音楽の演奏は中断される。

●サンプルプログラム

```
10 CLS
20 LOCATE10,10:PRINT"フ ル サ ト"
30 PLAY "T80S0M8000V8","T80S0M8000V8"
40 PLAY "L4O4","L4O3"
50 PLAY "GGG","GBO4D"
60 PLAY "A.L8BA","O3F+AO4D"
70 PLAY "B4B4O5C","O3G4B4O4D"
80 PLAY "D2R4","O3GBO4D"
90 PLAY "L4CDE","O3EGO4C"
100 PLAY "O4B.L8O5CO4L4B","O3GBO4D"
110 PLAY "AAF+","O3F+AL8O4DC"
120 PLAY "G2R4","L4O3BO4DO3G"
130 END
```

ＳＯＵＮＤ（サウンド）

■効果音を発生させる

ＰＳＧのレジスタに直接データを書き込んで音を出す。

書式 ＳＯＵＮＤ　レジスタ番号，データ

説明　ＭＳＸのＰＬＡＹやＢＥＥＰなどで出す音は、すべてＰＳＧ（プログラマブルサウンドジェネレータ）と呼ばれるレジスタの働きで音を出しているが、このＰＳＧのレジスタに直接データを入力して音を出す命令がＳＯＵＮＤである。

この命令は、こまかい指定ができるので､いろいろな音を出せるが､ＰＬＡＹ文のように簡単に音は出ないので詳しい説明は省略する。

書式例　ＳＯＵＮＤ　7,55：ＳＯＵＮＤ　8,8　：ＳＯＵＮＤ 0, 255

ＰＳＧに用意されているレジスタとその働きは次のとおり。

Ｒ０，Ｒ１	チャンネルＡの周波数を指定する。
Ｒ２，Ｒ３	チャンネルＢの周波数を指定する。
Ｒ４，Ｒ５	チャンネルＣの周波数を指定する。
Ｒ６	ノイズの周波数を指定する。
Ｒ７	各チャンネルから音を出すかどうかを指定する。
Ｒ８，Ｒ９，Ｒ10	チャンネルＡ，Ｂ，Ｃのそれぞれの音量を指定する。
Ｒ11，Ｒ12	エンベロープ（ＰＬＡＹ文と同じ）の周期を指定する。
Ｒ13	エンベロープのパターンを指定する。

（例）
```
10 SOUND 6, 15:SOUND 7, 7:SOUND 8, 16
20 SOUND 9, 16:SOUND 10, 16:SOUND 11, 0
30 SOUND 12, 10:SOUND 13, 1
```

上の例は、ピストルの発射音を出すプログラムである。

ＢＥＥＰ（ビープ）

■スピーカを鳴らす

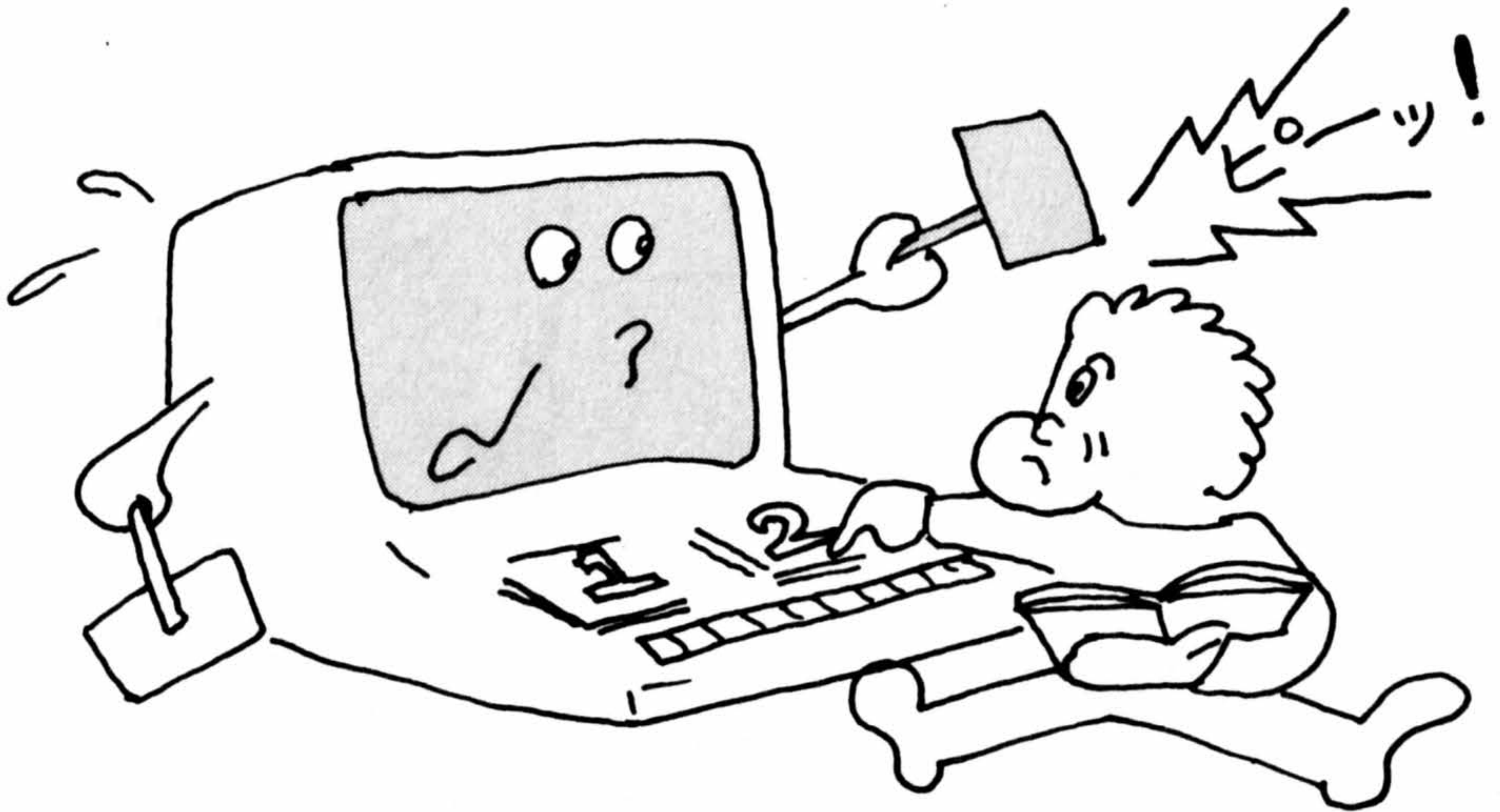
スピーカからピーッという音を出させる。

書式 ［ＢＥＥＰ］

説明 テレビのスピーカからピーッというブザー音を約0.04秒間鳴らす。

ＢＥＥＰ命令はコントロールコードの７（ＰＲＩＮＴ　ＣＨＲ＄（７）；）を実行するのと同じである。

（例）
```
10 S=STRIG(0)
20 IF S=0 GOTO 10
30 BEEP
40 GOTO 10
```

これは、スペースキーを押している間ＢＥＥＰで音を鳴らしているプログラムである。30行のＢＥＥＰを、ＰＲＩＮＴ　ＣＨＲ＄（７）；と書き換えても同じ動作をする。

SET VIDEO（セットビデオ）

■スーパーインポーズやデジタイズのモード指定

このＢＡＳＩＣ命令を使用しスーパーインポーズを行なうためには、専用のハードウェアが必要である。

スーパーインポーズやデジタイズなどのモードを指定したり、入力したい映像や音声の信号を切り換える命令である。

書式 ｜ＳＥＴ　ＶＩＤＥＯ ２，（輝度），０，１，，（ビデオ入力）｜

説明 輝度は、通常０を指定する。ビデオに比べて画像が暗い場合、１を指定すると文字が明るく表示され、見やすい画面になる。

ビデオ入力は、外部入力ビデオ記号で、０ならばＲＧＢ、１ならばビデオ信号端子をそれぞれ選択する。

スーパーインポーズの状態から通常の状態に戻るときは、ぜんぶ０を指定する。

ＳＥＴ　ＶＩＤＥＯ　０，０，０，０

⑥割り込み処理

ON SPRITE GOSUB (オン スプライト ゴーサブ)

■スプライトが衝突したら割り込み処理

スプライトが重なって表示されたときに実行する割り込み処理ルーチンの開始行番号を指定する。

書式 ON SPRITE GOSUB 行番号

説明 PUT SPRITEで表示したスプライトが、他のスプライトと重なったとき、SPRITE ONの状態なら、現在実行しているプログラムを中断して、ON SPRITE GOSUBで指定した行番号を実行する。

行番号は、割り込み処理ルーチンを開始する行番号である。2つのスプライトの座標値が違っていても、スプライトの一部が重なり合えば割り込みがかかる。

ただし、割り込み処理ルーチンの実行中は自動的にSPRITE STOPの状態になるので、スプライトが重なって表示されても割り込みは保留される。

割り込み処理ルーチンからもとに戻るときは、GOSUB文と同じようにRETURN命令を使用する。RETURN文に行番号が指定されていなければ、割り込みがかかったときに中断したプログラムが再開され、行番号が指定されていれば、その行番号からプログラムが再開される。

RETURN文に行番号を指定していないとき、再開するプログラムは、PUT SPRITEのつぎのプログラムとは限らない。これは、PUT SPRITE文で他のスプライトに重ねて表示しても、同時に割り込み処理ルーチンにジャンプするとは限らず、PUT SPRITEの後の命令がいくつか実行されてしまうからである。

割り込み処理ルーチンの中で、SPRITE OFFが実行されていなければ、RETURN命令の実行とともにSPRITE STOPの状態は解除さ

れ、ふたたびＳＰＲＩＴＥ　ＯＮの状態に戻る。

書式例　　ＯＮ　ＳＰＲＩＴＥ　ＧＯＳＵＢ　100

上の例は、スプライトが重なって表示されたときに現在実行しているプログラムを中断し、100 行からの割り込み処理ルーチンを実行する。

(関連する命令)

ＳＰＲＩＴＥ　ＯＮ

この命令はスプライトが衝突したら割り込みを許可する。この命令の実行後、スプライトが重なって表示されるとＯＮ　ＳＰＲＩＴＥ　ＧＯＳＵＢで指定した行番号の割り込み処理ルーチンを実行する。

ＳＰＲＩＴＥ　ＯＦＦ

この命令は、スプライトが衝突しても割り込みを禁止する。この命令を実行後、スプライトが重なって表示されても割り込みはかからない。

ＳＰＲＩＴＥ　ＳＴＯＰ

この命令は、スプライトが衝突しても割り込みを保留する。この命令を実行後、スプライトが重なって表示されても、その時点では割り込み処理をせず、つぎにＳＰＲＩＴＥ　ＯＮ　が実行されると割り込み処理ルーチンを実行する。ただしこの命令は、ＳＰＲＩＴＥ　ＯＮの状態で実行しないと無効になる。

(例)
```
10  ON SPRITE GOSUB 100
20  SCREEN 1, 0
30  SPRITE$(0)=STRING$(8, CHR$(255))
40  PUT SPRITE 0,(160 , 100), 2, 0
50  SPRITE ON
60  X=X+8
70  PUT SPRITE 1,(X, 100), 8, 0
80  FOR I=1 TO 100 :NEXT
90  GOTO 60
100  X=0
110  RETURN 70
```

このプログラムは、画面に緑と赤の２つの箱が表示され、赤い箱は緑の箱の

ある方向に向かって動いていき、２つの箱が重なると赤い箱はもとの場所に戻されてふたたび緑の箱に向かって動くという動作を繰り返す。

２つの箱がスプライトで表示されており、２つのスプライトが重なって表示されたときに割り込みがかかり、100 行からのプログラムを実行する。変数Ｘが０になるので赤い箱がもとに戻る。

注

１　ＯＮ　ＳＰＲＩＴＥ　ＧＯＳＵＢとＳＰＲＩＴＥ　ＯＮ／ＯＦＦ／ＳＴＯＰは必ずペアで使用する。

２　この割り込み処理は、ＢＡＳＩＣプログラムの実行中以外は行なわない。

３　ＰＵＴ　ＳＰＲＩＴＥでスプライトを重ねて表示したと同時に割り込み処理ルーチンにジャンプするとは限らない。たとえば、（例）のプログラムの80行を消してＲＵＮすると、赤い箱が緑の箱にぶつかって通り過ぎてから割り込み処理ルーチンが実行されることがある。スプライトが重なって表示されてから、割り込み処理ルーチンが実行されるまでに70行のプログラムが何回か実行されているということがわかる。

４　ＯＮ　ＥＲＲＯＲ　ＧＯＴＯやＯＮ　ＫＥＹ　ＧＯＳＵＢあるいはＯＮ　ＳＴＯＰ　ＧＯＳＵＢで定義した割り込み処理ルーチンの実行中は、スプライトが衝突しても割り込みは保留される。

ON STOP GOSUB (オン ストップ ゴーサブ)

■ストップキーで割り込みをかける

CTRL+STOPキーを押したときに実行する割り込み処理ルーチンの開始行番号を指定する。

書式 ON STOP GOSUB 行番号

説明 CTRL（コントロール）キーとSTOPキーが同時に押されたとき、STOP ONの状態なら、現在実行しているプログラムを中断して、ON STOP GOSUBで指定した行番号を実行する。ただし、STOPキーだけを押しても割り込みはかからない。

この命令は主に、プログラムの実行中に用ができたりしても、そこでプログラムを中断させたくないときや誤ってSTOPキーを押してプログラムを中断させるのを防止するようなときに使用する。

割り込み処理ルーチンの実行中は、自動的にSTOP STOPの状態になるので、CTRL+STOPキーが押されても、割り込みは保留される。

割り込み処理ルーチンからもとに戻るときは、GOSUB文と同じようにRETURN命令を使用する。RETURN文に行番号を指定していなければ、割り込みがかかったとき中断したプログラムが再開され、行番号を指定していれば、その行番号からプログラムの実行が開始される。

割り込み処理ルーチンの中で、STOP OFFが実行されていなければ、RETURN命令の実行とともにSTOP STOPの状態は解除され、ふたたびSTOP ONの状態に戻る。

書式例 ON STOP GOSUB 100

上の例はCTRLキーとSTOPキーが同時に押されたときは、現在実行しているプログラムを中断して100 行からの割り込み処理ルーチンを実行する。

（関連する命令）

STOP ON

この命令はCTRL+STOPキーが押されたとき、割り込みを許可する。この命令の実行後、CTRLキーとSTOPキーを同時に押すと、ON STOP GOSUBで指定した行番号の割り込み処理ルーチンを実行する。

STOP OFF

この命令はCTRLキーとSTOPキーが押されても、割り込みを禁止する。この命令を実行後、CTRLキーとSTOPキーを同時に押しても割り込みはかからない。

STOP STOP

この命令は CTRLキーと STOPキーを同時に押しても割り込み保留する。この命令の実行後、 CTRLキーと STOPキーを同時に押してもその時点では割り込み処理せず、つぎにSTOP ONが実行されると割り込み処理ルーチンが実行される。ただし、この命令が有効なのは、STOP ONの状態のときである。

（例）
```
10  ON STOP GOSUB 100
20  CLS
30  STOP ON
40  PRINT "*";
50  A$=INKEY$
60  IF A$="E" THEN END
70  GOTO 40
100  PRINT
110  PRINT "CAN NOT STOP"
120  FOR 1=I TO 1000:NEXT
130  RETURN
```

```
***********  ***
*********
CAN NOT STOP
***********  ***
```

前ページのプログラムは、＊マークがつぎつぎに表示されていくが、CTRLキーとSTOPキーを押してプログラムを止めようとしても止まらず、ふたたび＊マークの表示を続ける。これはCTRLキーとSTOPキーを同時に押したため割り込みがかかり、100 行からの割り込み処理ルーチンを実行してもとのプログラムの実行を再開するため止まらなくなっている。

止める時は、Eキーを押す。

注

1 ON STOP GOSUBと STOP ON／OFF／STOPは必ずペアで使用する。

2 この割り込み処理ができるのは、BASICプログラムの実行中だけである。

3 この命令を実行すると、本来のCTRL＋STOPキーの機能（プログラムを中断する機能）が失われてしまう。リセットボタンを押すか電源を切るかしないとプログラムが止まらなくなってしまう。（例）のようにプログラムを中断するプログラムを中に組み込むか、バグのないプログラムに使用する。

4 ON ERROR GOTOやON KEY GOSUBで定義した割り込み処理ルーチンの実行中は、CTRL＋STOPキーでの割り込みは保留される。

ON STRIG GOSUB
(オン スティックトリガ ゴーサブ)

■トリガボタンを使って割り込む

ジョイスティックのトリガボタンを押したときに実行する割り込み処理ルーチンの開始行番号を指定する。

書式 ON STRIG GOSUB 行番号1，行番号2，………

説明 ジョイスティックのトリガボタンまたはキーボードのスペースキーが押されたとき、STRIG ONの状態なら、現在実行しているプログラムを中断して、ON STRIG GOSUBで指定した行番号のプログラムを実行する。

行番号は最大5個まで指定することができ、最初の行番号から順番につぎのトリガボタンに対応する。

1番目： スペースキー

2，4番目： ポート1につながるジョイスティックのトリガボタン

3，5番目： ポート2につながるジョイスティックのトリガボタン

ジョイスティック1個に2つの行番号が指定できるのは、ジョイスティックに2つのトリガボタンがあるからである。

ふつうは、スティックについているトリガボタンが、2または3に対応し、台についているトリガボタンが4、5に対応している。ただし、ジョイスティックによっては、トリガボタンがひとつしかないものや、スティックと台のトリガボタンが中でつながっているものもあるため、自分のジョイスティックがどのタイプかを調べてから使用する。

使用しないトリガボタンまたはスペースキーに対応する行番号の指定は省略してもかまわない。

割り込み処理ルーチンの実行中は、自動的にSTRIG STOPの状態になり、トリガボタンが押されても割り込みは保留される。

割り込み処理ルーチンからもとに戻るときは、GOSUB文と同じようにRETURN命令を使用する。

ＲＥＴＵＲＮ文に行番号を指定していなければ、割り込みがかかったときに、中断したプログラムが再開され、行番号を指定しているときは、その行番号からプログラムの実行が開始される。

割り込み処理ルーチンの中でＳＴＲＩＧ　ＯＦＦが実行されていなければ、ＲＥＴＵＲＮ命令の実行とともにＳＴＲＩＧ　ＳＴＯＰの状態は解除され、ふたたびＳＴＲＩＧ　ＯＮの状態に戻る。

書式例　ＯＮ　ＳＴＲＩＧ　ＧＯＳＵＢ　，1000

ポート１につながるジョイスティックのトリガボタンが押されたときは、プログラムを中断して1000行からの割り込み処理ルーチンのプログラムを実行する。

（関連する命令）

ＳＴＲＩＧ（ｎ）　ＯＮ

ｎは０～４の数値である。この命令は、トリガボタンが押されたとき割り込みを許可する。この命令の実行後、トリガボタンを押すとＯＮ　ＳＴＲＩＧ　ＧＯＳＵＢで指定した行番号の割り込み処理ルーチンが実行される。

ＳＴＲＩＧ（ｎ）　ＯＦＦ

この命令は、トリガボタンでの割り込みを禁止する。この命令の実行後、トリガボタンが押されても割り込みはかからない。

ＳＴＲＩＧ（ｎ）　ＳＴＯＰ

この命令は、トリガボタンでの割り込みを保留する。この命令の実行後、トリガボタンが押されても、その時点では割り込み処理をせず、つぎにＳＴＲＩＧ（ｎ）　ＯＮが実行されると割り込み処理ルーチンが実行される。

ただし、この命令が有効なのは、ＳＴＲＩＧ（ｎ）　ＯＮの状態のときだけである。

割り込み

nの0～4に対応するトリガボタンはつぎのとおり。

0：スペースキー

1，3：　ポート1につながるジョイスティックのトリガボタン

2，4：　ポート2につながるジョイスティックのトリガボタン

（例）
```
10 ON STRIG GOSUB 100
20 STRIG(0) ON
30 PRINT "MSX"
40 GOTO 30
100  BEEP
110  COLOR,8
120  FOR I=1 TO 500:NEXT
130  COLOR,4
140  RETURN
```

```
MSX
MSX
MSX
MSX
MSX
MSX
```

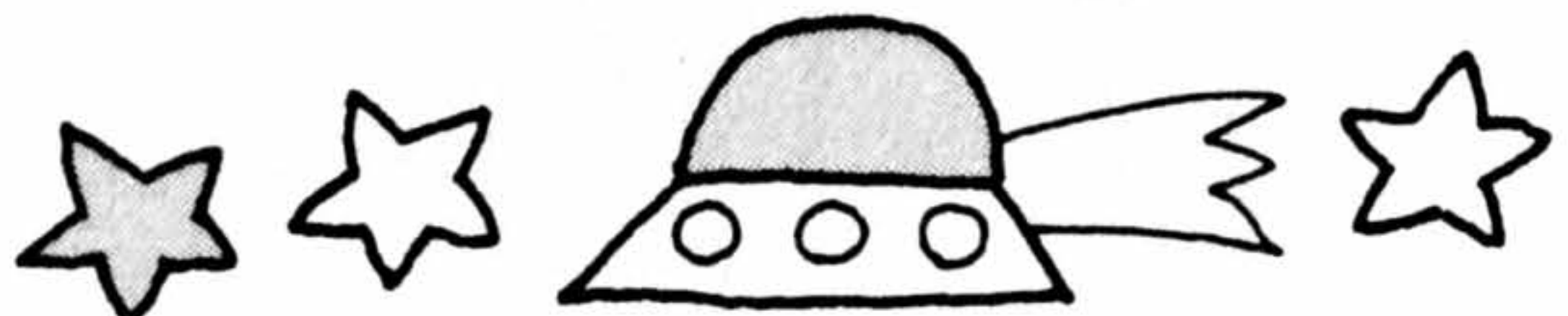
"MSX"という文字をつぎつぎに表示していくが、スペースキーを押すとBEEP音が鳴り、一瞬画面が赤くなる。スペースキーを押したためにプログラムの実行が中断され、100 行からのプログラムが実行されたからである。

ジョイスティックを使用するときは、ポート1につながっているなら、10行をON STRIG GOSUB ,100 と直し、20行のカッコの中を1にする。

注

1 ON STRIG GOSUBと STRIG ON／OFF／STOPは必ずペアで使用する。

2 この割り込みができるのは、BASICプログラム実行中である。

3 ON ERROR GOTOやON KEY GOSUBあるいはON STOP GOSUB、ON SPRITE GOSUBで定義した割り込み処理ルーチンの実行中は、トリガボタンを使用した割り込みは保留される。

ON INTERVAL GOSUB（オン　インターバル　ゴーサブ）

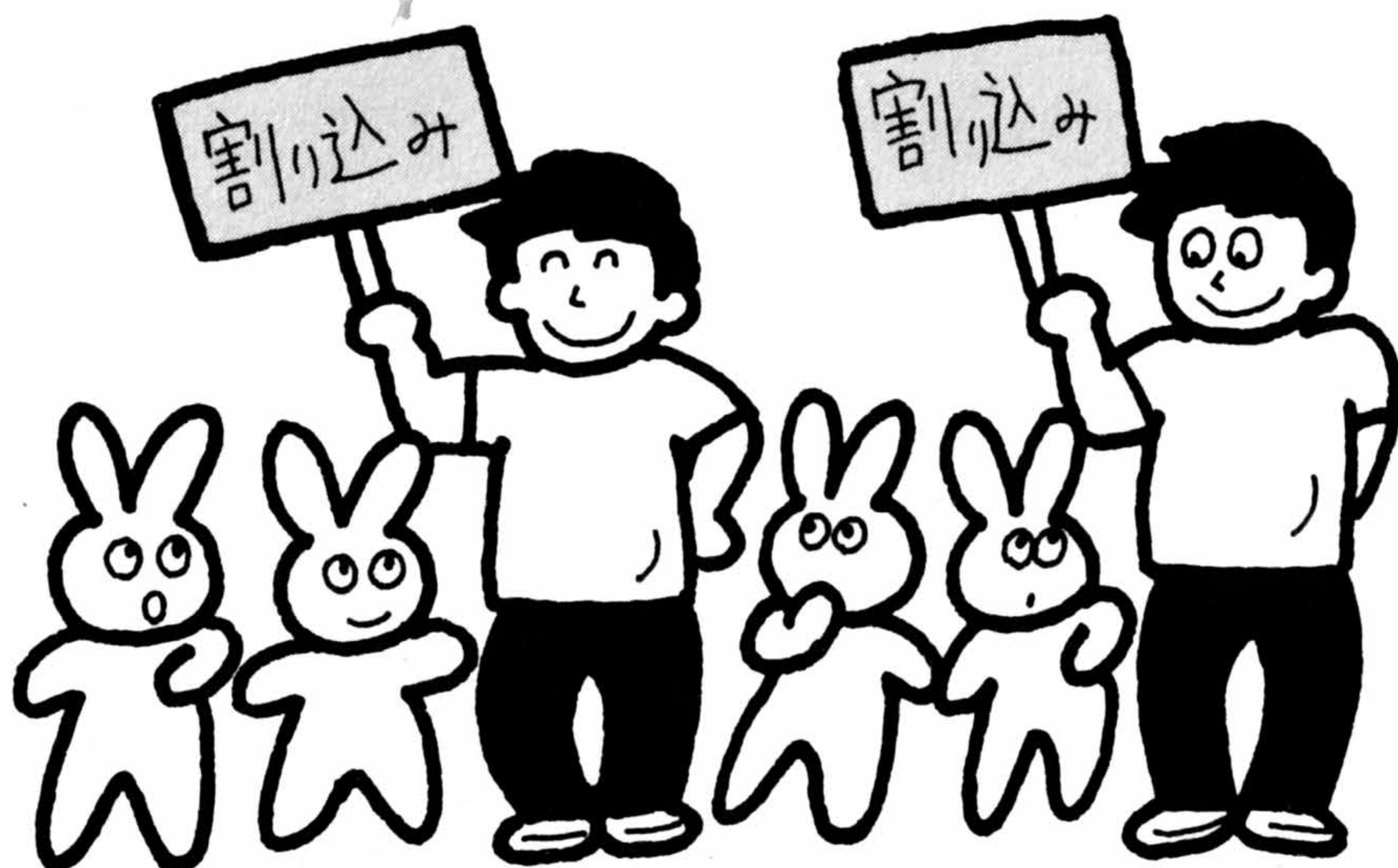
■一定の時間間隔で割り込む

　この命令は割り込みの時間間隔を設定し、割り込みがかかったときの処理ルーチンの開始行を指定する。

書式　ON　INTERVAL＝時間　GOSUB　行番号

説明　時間の単位は1/60秒である。たとえば1 秒ごとに割り込みをかけるなら60（=60/60秒）とする。指定した時間の間隔でタイマー割り込みが発生する。ＩＮＴＥＲＶＡＬ　ＯＮの状態なら、現在実行しているプログラムを中断し、ＯＮ　ＩＮＴＥＲＶＡＬ　ＧＯＳＵＢで指定した行番号のプログラムを実行する。

　割り込み処理ルーチンの実行中は、自動的にＩＮＴＥＲＶＡＬ　ＳＴＯＰの状態になり、割り込みは保留される。

　割り込み処理ルーチンからもとに戻るときは、ＧＯＳＵＢと同じようにＲＥＴＵＲＮ命令を使用する。ＲＥＴＵＲＮ文に行番号を指定していなければ、割

り込みがかかったときに中断したプログラムが再開され、行番号を指定していれば、その行番号からプログラムの実行が開始される。

割り込み処理ルーチンの中でＩＮＴＥＲＶＡＬ　ＯＦＦが実行されていなければ、ＲＥＴＵＲＮ命令の実行とともにＩＮＴＥＲＶＡＬ　ＳＴＯＰの状態は解除され、ふたたびＩＮＴＥＲＶＡＬ　ＯＮの状態に戻る。

書式例　ＯＮ　ＩＮＴＥＲＶＡＬ＝120　ＧＯＳＵＢ　1000

上の例は　2 秒ごとに1000行からの割り込み処理ルーチンを実行する。

（関連する命令）

ＩＮＴＥＲＶＡＬ　ＯＮ

この命令はタイマーの割り込みを許可する。命令の実行後、ＯＮ　ＩＮＴＥＲＶＡＬ　ＧＯＳＵＢで設定した時間ごとに割り込みがかかり、指定した行番号からの割り込み処理ルーチンを実行する。

ＩＮＴＥＲＶＡＬ　ＯＦＦ

この命令はタイマー割り込みを禁止する。命令の実行後、タイマー割り込みはかからない。

ＩＮＴＥＲＶＡＬ　ＳＴＯＰ

この命令はタイマー割り込みを保留する。命令の実行後は、指定した時間では割り込み処理をせず、つぎのＩＮＴＥＲＶＡＬ ＯＮが実行されると割り込み処理ルーチンが実行される。

ただし、この命令はＩＮＴＥＲＶＡＬ　ＯＮの状態のときだけ有効である。

（例）
```
10  ON INTERVAL=60 GOSUB 100
20  CLS:TIME=0
30  INTERVAL ON
40  GOTO 40
100  LOCATE 0,10
110  T=INT(TIME/60)
120  PRINT T;"ビョウケイカ"
130  RETURN
```

```
2ビョウケイカ
```

40行にはＧＯＴＯ　40という命令があり、ここが無限ループとなっているため、１度この命令を実行すると，プログラムは40行から抜け出ることはできない。１秒ごとに画面に表示する時間が更新されていく。これは、１秒ごとにプログラムの実行が中断されて、100 行からの割り込み処理ルーチンが実行されるからである。

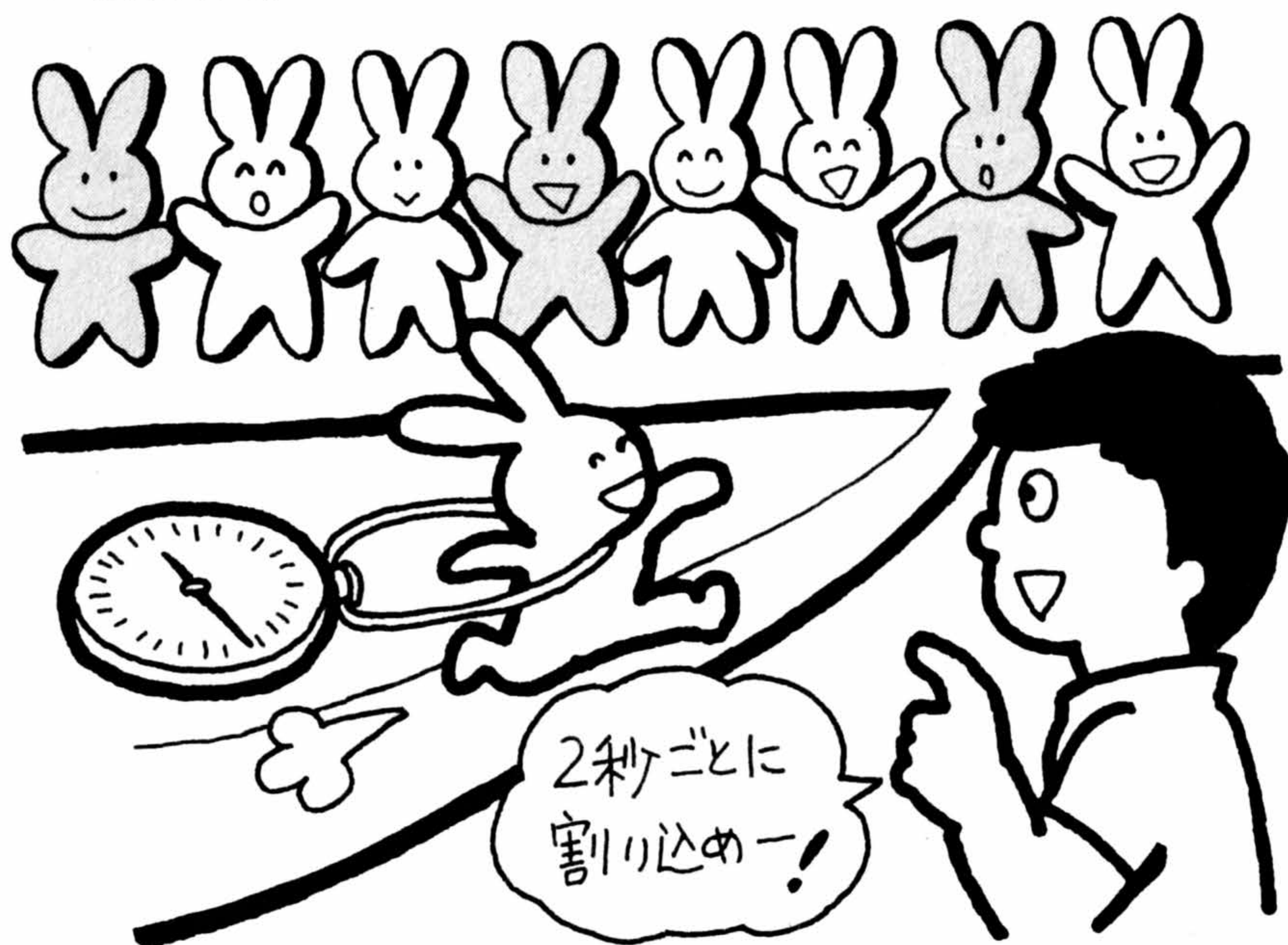
注

1　ＯＮ　ＩＮＴＥＲＶＡＬ　ＧＯＳＵＢとＩＮＴＥＲＶＡＬ　ＯＮ／ＯＦＦ／ＳＴＯＰは必ずペアで使用する。

2　この割り込みができるのは、ＢＡＳＩＣプログラムの実行中だけである。

3　割り込みのタイマーとＴＩＭＥ関数の値は関係がない。

4　ＯＮ　ＩＮＴＥＲＶＡＬ　ＧＯＳＵＢ文は、割り込みを定義するだけで、実行しても割り込みはかからない。

5　他の全ての割り込み処理ルーチンを実行している間は、タイマー割り込みは保留される。

ON KEY GOSUB(オン キー ゴーサブ)

■ファンクションキーで割り込みをかける

ファンクションキーが押されたとき実行する割り込み処理ルーチンの開始行を指定する。

書式 ON KEY GOSUB 行番号1，行番号2，……………

説明 ファンクションキーFnが押されたとき、KEY（n） ONの状態なら、現在実行しているプログラムを中断して、ON KEY GOSUBで指定した左側からn番目の行番号が実行される。nはファンクションキーの番号である。

MSXのファンクションキーはF1～F10まであるから、行番号は最大10個まで指定できる。行番号1はF1キー、行番号2はF2キー………行番号10はF10キーである。使用しないファンクションキーに対応する行番号の指定は省略できる。

割り込み処理ルーチンの実行中は、自動的にKEY（n） STOPの状態になるので、割り込みは保留される。

割り込み処理ルーチンからもとに戻るときは、GOSUB文と同じようにRETURN命令を使用する。RETURN文に行番号を指定していなければ、割り込みがかかったときに中断したプログラムが再開され、行番号を指定していれば、その行番号からプログラムの実行が開始される。

割り込み処理ルーチンの中で、KEY（n） OFFが実行されていなければ、RETURN命令の実行とともにKEY（n） STOPの状態は解除されふたたびKEY（n） ONの状態に戻る。

書式例 ON KEY GOSUB 100,200,300

ファンクションキーF1が押されたときは100行、F2が押されたときは200行、F3が押されたときは300 行からの割り込み処理ルーチンを実行する。

(関連する命令)

KEY（n） ON

この命令はファンクションキーが押されたときの割り込みを許可する。命令の実行後、ファンクションキーを押すと、ON　KEY　GOSUBで指定した行番号の割り込み処理ルーチンが実行される。nはファンクションキーの番号である。

KEY（n）　OFF

この命令はファンションキーを使った割り込みを禁止する。命令の実行後は、ファンクションキーが押されても割り込みはかからない。

KEY（n）　STOP

この命令はファンクションキーを使った割り込みを保留する。命令の実行後にファンクションキーが押されても割り込み処理はせず、つぎにKEY（n）が実行されると割り込み処理ルーチンが実行される。

ただし、この命令が有効なのはKEY（n）　ONの状態のときだけである。

（例）
```
10  ON  KEY  GOSUB  100 ,200 ,300
20  KEY（1）  ON：KEY（2）  ON：KEY（3）  ON
30  CLS
40  LOCATE  0, 10
50  PRINT  "PUSH  F1－F3  KEY"
60  GOTO  60
100  BEEP：COLOR  , 2
110  RETURN
200  BEEP：COLOR  , 8
210  RETURN
300  BEEP：COLOR  , 10
310  RETURN
```

```
PUSH  F1～F3  KEY
```

上のプログラムは、ファンクションキーのF１を押すと背景色が緑になる。F２キーを押すと赤、F３キーを押すと黄色に変わる。

このプログラムは、60行が無限ループとなっているため、１度60行を実行すると、そこから抜けることができない。ファンクションキーを押すと割り込み

がかかり、そのファンクションキーに対応する割り込み処理ルーチンが実行されて背景色が変わる。

注

1　ＯＮ　ＫＥＹ　ＧＯＳＵＢとＫＥＹ（ｎ）　ＯＮ／ＯＦＦ／ＳＴＯＰは必ずペアで使用する。

2　この割り込みができるのは、ＢＡＳＩＣプログラムの実行中だけである。

⑦ 関　　数

RND（ランダム）

乱数を発生させる

ＲＮＤ関数は乱数(不規則な数)を発生させる。

書式 ＲＮＤ（ｎ）

ＲＮＤ（１）　０以上１未満の乱数を発生させる。

説明　発生する乱数はｎの値によって次のようになる。

ｎが正のときは、０以上１未満の乱数を発生させる。発生する乱数はＲＵＮを実行するごとに同じ系列である。

(例)
```
10 FOR I=1 TO 5
20 PRINT RND(1)
30 NEXT
```

ＲＵＮさせた結果をみれば、ＲＮＤ（１）は同じ数を同じ順番で発生させていることがわかる。

ｎが０の時は、１つ前に発生した乱数を発生させる。すなわち、すぐ前に実行されたＲＮＤの値と同じ値が発生するということである。

ｎが負の時は、乱数の発生を初期化する。ただし、この時はその負の値によって固有の乱数系列をつくるだけで、乱数を発生させることはできない。

そこでｎの値を変えるために、ＴＩＭＥ関数を利用して、ＲＮＤ(－ＴＩＭＥ)とすれば毎回発生する乱数を変えることができる。

書式 ＲＮＤ（－ＴＩＭＥ）

実行するたびに乱数系列を変える。(乱数として使うこともできる。)

(例)
```
10 A=INT (RND (-TIME ) *10 )
20 INPUT "PUSH 0-9 KEY";B
30 IF A=B THEN PRINT "アタリマシタ" ELS
   E PRINT "ハズレマシタ"
```

```
40 GOTO 10
```

```
PUSH 0-9 KEY? 5
アタリマシタ
PUSH 0-9 KEY? 8
ハズレマシタ
```

これは簡単な数あてゲームのプログラムである。

ここでは、10行で発生した0～9までの乱数と20行で入力した数が同じならばアタリである。

10行目は、RND(－TIME) で発生した0以上1未満の数を発生させ、その値に10をかけることによって0以上10未満の数を発生させている。

さらに、INT命令を使ってその値を整数にし、0～9までの整数を発生させている。

このようにRNDは、INTとあわせて使われることが多い。

注

1　RNDのカッコ内が正の数の時は、数に関係なく発生する乱数は同じであるため、一般的には中に1を入れる。

2　RNDで発生する値は0以上1未満で、1は含まれていない。

DIM(ディメンション)

配列変数の次元と最大添え字数指定

配列変数の次元とその変数の添え字の最大値を指定して、その変数に対してメモリ領域を確保する。

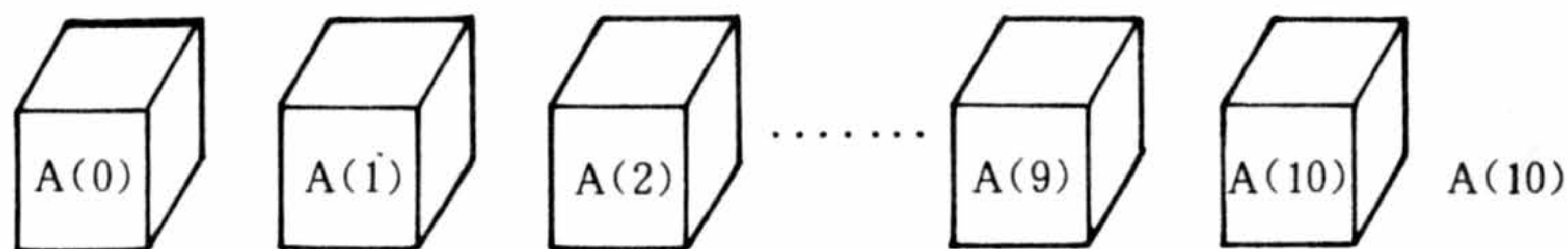
書式　DIM　変数名（添え字の最大値，添え字の最大値，……）

説明　配列変数の添え字の最大値を設定し、その配列に必要なメモリをあらかじめ確保しておく。

カッコの中の添え字の最大値の個数が、配列の次元数を示す。

いいかえれば配列とは、あらかじめ数値または文字を格納するための箱を用意する命令と考えるとわかりやすい。

たとえばＤＩＭ　Ａ（10）を図に示すと次のようになる。

A(0)　A(1)　A(2)　……　A(9)　A(10)　A(10)

つまりＤＩＭ　Ａ（10）とは、Ａ（0）～Ａ（10）の11個の箱（記憶場所）を用意することである。

●ＤＩＭ２次元配列命令

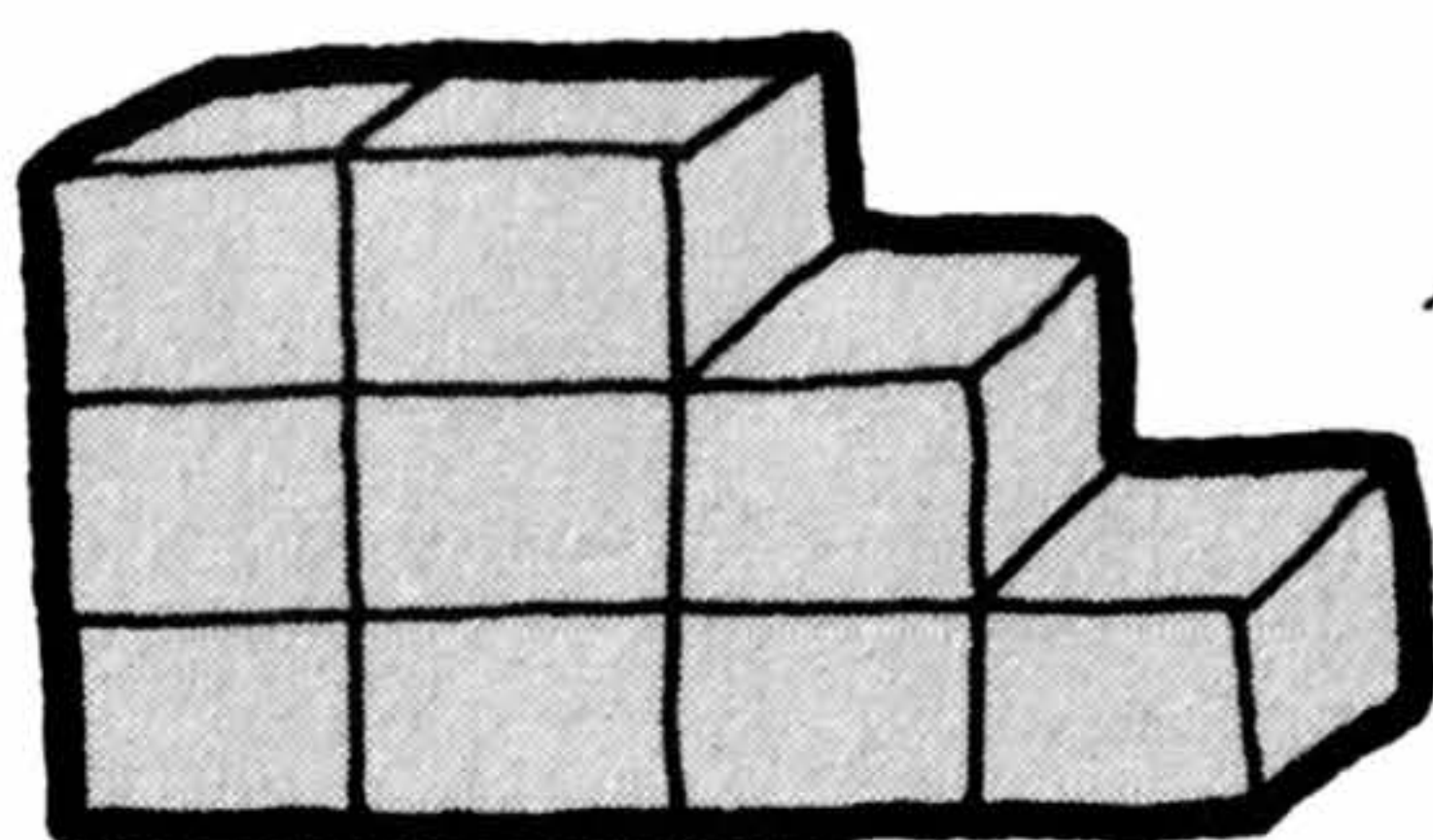
ＤＩＭ命令にはこのほかにＮ次元配列がある。ここでは２次元配列を例にとって説明する。２次元配列Ａ（4,4）を図に示すと、下図のようになり、合計25個の箱を用意したことになる。変数名は省略している。

ただし、ＤＩＭ文を宣言しないで配列変数を使用したときは、その添え字の最大値は10が設定されるので、10以下の場合は配列変数の宣言を省略することができる。

ＤＩＭ文が実行されると、指定された配列のすべての要素の値は、数値変数のときは0に、文字変数のときは"ヌルストリング"に設定される。

書式例　ＤＩＭ　Ａ（10，20）

２次元配列Ａを設定する。

(0，0)	(0，1)	(0，2)	(0，3)	(0，4)
(1，0)	(1，1)	(1，2)	(1，3)	(1，4)
(2，0)	(2，1)	(2，2)	(2，3)	(2，4)
(3，0)	(3，1)	(3，2)	(3，3)	(3，4)
(4，0)	(4，1)	(4，2)	(4，3)	(4，4)

書式例　ＤＩＭ　Ａ(10，20)

２次元配列Ａを設定する。

(例)

```
10 DIM B$(3)
20 B$(1)="ファースト"
30 B$(2)="セカンド"
40 B$(3)="サード"
50 FOR I=1 TO3
60 PRINT B$(1)
70 NEXT
80 END
```

```
ファースト
セカンド
サード
```

Ｂ＄(1)〜Ｂ＄(3)までに設定した文字列を、50〜70行のＦＯＲ〜ＮＥＸＴを使用して画面に表示させている。

10行目は添え字の最大値が３の文字型の配列変数Ｂ＄を宣言している。

注

1　指定した添え字の最大値より大きい値を指定すると、Subscript out of rangeのエラーになる。

2　配列変数はメモリを占有するため、ＤＩＭ文を使うときはメモリが不足しないことを確認しておく。(確認するときは、ＰＲＩＮＴ　ＦＲＥ(0)と入力すれば、残りのメモリのバイト数がわかる)

3　いちどＤＩＭ文で定義した配列変数は、ＣＬＥＡＲまたはＥＲＡＳＥ命令を実行しない限り再定義はできない。

ＳＷＡＰ(スワップ)

■２個の変数を入れ換える

指定した２つの変数の値を交換する。

書式　ＳＷＡＰ　変数１，変数２

説明　変数１と変数２の値を交換する。

変数の型は、整数型、単精度型、倍精度型、文字型、配列変数のいずれであっても交換することができるが、２つの変数の型はかならず同じでなくてはな

らない。（一致していないときは、"Type mismatch" エラーになる）

書式例　SWAP　A$，B$

A$とB$の値を交換する。

（例）

```
10 INPUT A, B
20 GOSUB 60
30 SWAP A, B
40 GOSUB 60
50 END
60 PRINT "A=";A, "B=";B
70 RETURN
```

```
? 10
?? 25
A= 10    B= 25
A= 25    B= 10
```

20行と40行で同じ60行からのサブルーチンを実行しているのにもかかわらず、表示された変数AとBの値が異なるのは、30行のSWAP命令によってAとBの値が交換されているためである。

FRE（フリー）

メモリの未使用領域の大きさを調べる

プログラム領域または文字領域のメモリの未使用領域の大きさを調べる。

書式

```
FRE（{0   }）
     {文字列}
```

説明　FRE（0）は、BASICのプログラム領域と変数領域の未使用領域の大きさを調べる。

カッコの中の0はダミーで、0以外の定数や変数でもかまわないが、一般には0を指定する。

FRE（文字式）は、文字領域（文字変数の内容を格納するメモリ）の大きさを調べる時に使用する。

この命令を実行すると、文字領域中の不要な文字列を消去して、必要な文字列だけを残して文字領域を整理する。これをガベージコレクションという。

書式例　FRE（0）

BASICの未使用領域を調べ、その値を関数の値として得る。

（例）　ＰＲＩＮＴ　ＦＲＥ（０），ＦＲＥ（"Ａ"）　…（1）

```
10 A$="MSX BASIC"
20 PRINT A$
30 END
```

まず（１）のプログラムを直接入力する。次に、プログラムを入力してＲＵＮさせてから、もう一度（1）のプログラムを入力する。

２回目に入力した時に表示された値は１回目の値よりも小さくなっているが、これはプログラムによってメモリが使用され、未使用領域が減ったためである。

ＤＥＦ　ＦＮ（デファイナブル ファンクション）

■関数を定義する

関数を定義して、それに名前を付ける。

書式　ＤＥＦ　ＦＮ　名前（変数，変数，………）＝定義式

説明　プログラム中でたびたび使う計算式と、その式で使う変数とを関数としてあらかじめ定義しておけば、その後のプログラム中でいちいち計算式を書かなくてもこの関数を呼び出すだけで計算ができる。この関数を定義するのが、ＤＥＦ　ＦＮである。

ＦＮのうしろに続く名前は、数値変数または文字変数の名前で、ＦＮ＋変数名が関数名となる。

変数は、定義式の中で使っているものと同じ名前の変数に対応する。つまり、定義式の中で何種類の変数を使っているかで、カッコの中の変数の数が決まるのである。

また、この変数は、定義式を計算する時に限り使われるので、プログラム中で同じ変数を使用しても影響はない。

呼び出し方は、ＤＥＦ　ＦＮ命令で定義した後に、プログラム中でＦＮ名前（定数，定数，……）とし、対応する変数に定数を代入した値が得られる。

書式　ＤＥＦ　ＦＮ　Ａ（Ｘ，Ｙ）＝３＊Ｘ＋４＊Ｙ＋２

３＊Ｘ＋４＊Ｙ＋２という計算式を、ＦＮ　Ａという関数として定義する。

（例）
```
10 DEF FNA（X，Y）=40*X+Y
20 B=FNA（4，7）
30 PRINT B
40 C=FNA（7，3）
50 PRINT C
60 END
```

```
167
283
```

40*X+Yという式をDEF　FN命令で関数として定義して、それを呼び出して計算を行なっているのが20行と40行である。

20行では40*4+7という計算が行なわれるため、変数Bには167という値が代入され、40行では40*7+3という計算が行なわれるため、変数Cに283という値が代入される。

注

1　計算式は必ず1行で書かなければならない。

2　DEF　FNで定義する関数は、数値関数、文字関数、その2つの混在している関数のいずれの型でもかまわないが、関数を呼び出すときに指定する定数とDEF　FNで指定した変数の型が同じでなければならない。

　また、関数名の型は数値、文字のどちらでもかまわないが、計算式の結果と同じ型でなければならない。

3　DEF　FN文は、それが定義される関数を呼び出す前に実行されなければならない。もし定義する前に関数が呼び出されると、Underfind user functionエラーとなる。

DEFINT／SNG／DBL／STR
（デファイナブル　インテンジャー　/シングル/ダブル/ストリング　）

■変数の型を宣言する

変数の型を整数、単精度、倍精度、文字の各型に宣言する。

書式　DEF型指定　文字の範囲

説明　書式にある「文字の範囲」で指定される文字で始まるすべての変数名の型を、「型指定」で示される型の変数として扱うように宣言する。

指定できる型は、次の4種類がある。

DEFINT——————変数を整数型に宣言する。

DEFSNG——————変数を単精度型に宣言する。

DEFDBL——————変数を倍精度型に宣言する。

DEFSTR——————変数を文字型に宣言する。

なにも宣言しない時は、変数はすべて単精度として扱われるが、ゲームなどでは小数点以下を必要とする計算はほとんどないので、変数をＤＥＦＩＮＴ文で整数型に宣言しておけば、計算速度が速くなり、ゲームのスピードをアップすることができる。また、ビジネス用のプログラムを作る時などは、有効桁数が6桁では不足してしまうことが多いが、その時はＤＥＦＤＢＬで倍精度型に直しておくと便利である。

ただし、変数が型宣言文字によって指定されていた時は、その型宣言文字がＤＥＦ文より優先される。

ＣＬＥＡＲ命令が実行されると、それまでのＤＥＦによる宣言はすべて無効となるので、変数はすべて単精度型変数として扱われる。

書式例　ＤＥＦＩＮＴ　Ａ

Ａで始まる変数（ＡＢ，ＡＦ……など）を整数型に宣言する。

ＤＥＦＳＮＧ　Ａ，Ｃ，Ｆ

Ａ，Ｃ，Ｆで始まる変数を単精度型に宣言する。

ＤＥＦＤＢＬ　Ａ－Ｚ

Ａ－Ｚまでの文字で始まる変数（つまりすべての変数）を倍精度型に宣言する。

ＤＥＦＳＴＲ　Ｃ，Ｆ－Ｈ

Ｃ，Ｆ，Ｇ，Ｈで始まる変数を文字型に宣言する。

（例）
```
10 DEFINT A
20 DEFSNG B
30 DEFDBL C
40 A=10/ 3
50 B=10/ 3
```

```
3
3.33333
3.3333333333333
```

```
60  C=10/3
70  PRINT  A
80  PRINT  B
90  PRINT  C
```

変数Aは整数型、Bは単精度型、Cは倍精度型に10〜30行で定義され、それぞれの変数に10÷3の計算結果を代入して表示させている。

この結果からもわかるように、DEF文によって変数の型を宣言すると有効桁数が変わる。

注1　DEFと型指定の間をあけることはできない。

LEFT$(レフトダラー)

■左側の文字列を指定してとり出す

書式　LEFT$(文字列，文字数)

説明　「文字列」の左側から「文字数」で指定した数だけの文字列をとり出して関数の値とする。

文字列は文字変数で指定してもかまわない。文字数は、式、変数、定数のいずれかで指定する。

もし指定した文字数が文字列の文字数より大きい場合は、文字数のすべてが関数の値となる。また、文字数を0としたとき、関数の値はヌルストリング(" ")となる。

書式例　LEFT$(A$，4)

文字変数A$の左側から4文字をとり出して関数の値とする。

(例)
```
10  A$="MSX  BASIC"
20  PRINT  LEFT$(A$, 3)
30  END
```

```
MSX
```

文字変数A$には、"MSX　BASIC"という文字列が代入されているが、20行目のPRINT文では、LEFT$によってA$の左側から3文字、すなわち"MSX"という文字だけをとり出して表示している。

注　文字列は 0～255 の範囲で指定する。

ＭＩＤ＄（ミドルダラー）

文字列中から指定した文字をとり出す

文字列中から指定した位置の文字を指定した文字数だけとり出す。

書式　ＭＩＤ＄（文字列，文字位置，文字数）

設明　「文字列」の指定した「文字位置」から、指定した「文字数」だけとり出して関数の値とする。

文字列は文字変数で指定してもよい。

文字位置と文字数は、式、変数、定数のいずれかで指定する。

文字位置から右側の文字数が指定した文字数より少ないとき、または文字数を省略したときは、右側のすべての文字列をとり出す。

文字列の文字数が、文字位置で指定した値より小さいときは、ヌルストリング（空白）になる。

書式例　ＭＩＤ＄(Ａ＄，４，２)

文字列Ａ＄の左から数えて４番目の文字から２文字をとり出して関数の値とする。

（例）
```
10  C$= "ABCDEFGH"
20  PRINT  MID$(C$, 4, 3)
30  END
```

```
DEF
```

文字変数Ｃ＄には、"ＡＢＣＤＥＦＧＨ"という文字列が代入されているが、20行目のＰＲＩＮＴ文では、ＭＩＤ＄によってＣ＄の左側の４文字目から３文字分、つまりＤＥＦという文字だけが表示される。

注

1　ＭＩＤ＄で指定する文字位置の値は 1～255 の範囲で、また文字数は 0～255 の範囲で指定する。

●ＭＩＤ＄の他の使い方

ＭＩＤ＄のもうひとつの使い方として、文字列の一部を他の文字列に置き換

えるということができる。

書式 MID$（文字列1，文字位置，文字数）＝文字列2

説明 文字列1の指定した文字位置から、指定した文字数の文字列を文字列2に置き換える。

文字列1、文字列2は文字変数で指定してもよい。

文字数は式、変数、定数のいずれかで指定する。

文字列1の指定した文字位置から右側の文字数が、指定した文字数より少ないときは、文字位置から右側すべてが置き換えられ、文字列2の余った部分は無視されるので、この命令を実行しても文字変数の長さが変わることはない。

文字数を省略したとき、または文字列2の文字数より多く文字数を指定したときは文字列2の文字がすべて置き換えられた後、不足分はそのままである。

（例）
```
10 A$="ABCDEFGH"
20 MID$(A$, 4, 3) = "OPQ"
30 PRINT A$
40 END
```

```
ABCOPQGH
```

A$には、"ABCDEFGH"という文字列が代入されているが、20行でA$の左から4番目の文字からの3文字が"OPQ"と置き換えられるため、A$の値は"ABCOPQGH"となる。

注

1　文字位置の値は、文字列1の文字数より大きい値で指定することはできない。また、0以下で指定することもできない。

2　文字列1はヌルストリングで指定することはできない。

RIGHT$(ライトダラー)

■右側の文字列を指定してとり出す

書式　RIGHT$（文字列，文字数）

説明　文字列の右側から文字数で指定した数だけ取り出して関数の値とする。

文字列は文字変数で指定してもかまわない。

文字数は式、変数、定数のいずれかで指定する。

指定した文字数が文字列の文字数より大きい場合は、文字数のすべてが関数の値となる。また、文字数を0としたときは、関数の値はヌルストリング（空白）となる。

書式例　RIGHT$（B$，3）

文字変数B$の右側から3文字をとり出して関数の値とする。

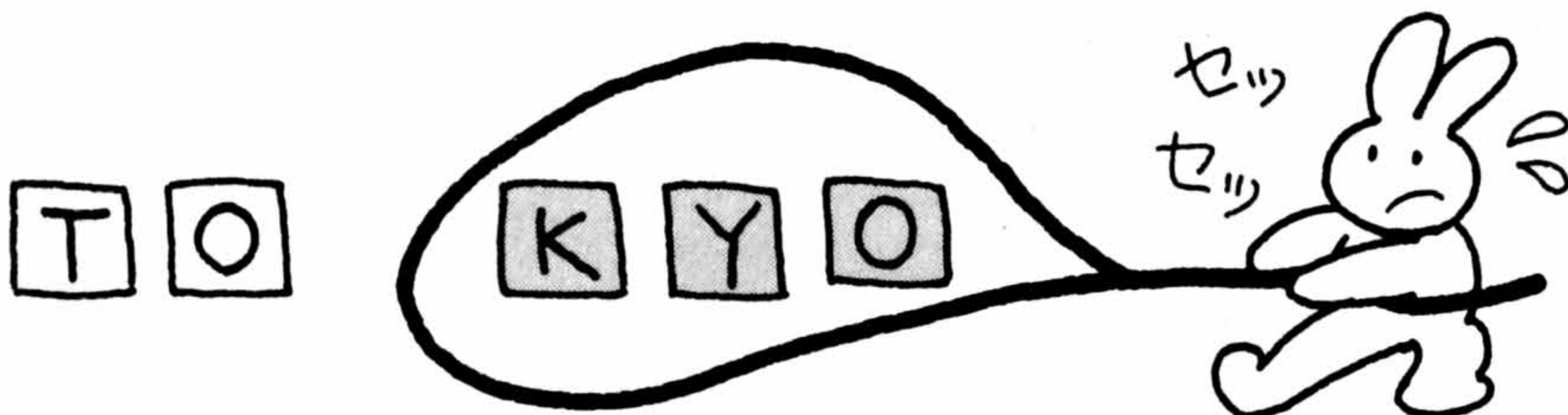
（例）

```
10 A$="CHIYODA  TOKYO"
20 PRINT  RIGHT$(A$, 5)
30 END
```

```
TOKYO
```

文字変数A$には"CHIYODA　TOKYO"という文字列が代入されているが、20行目のPRINT文では、RIGHT$によって指定されたA$の右側から5文字分、つまりTOKYOという文字だけが表示される。

注

1　文字数は 0〜255 の範囲で指定する。

ＣＬＥＡＲ（クリア）

■変数を初期化する

現在使用されている変数をすべて初期化し、オープンされているファイルをすべてクローズする。

書式 ＣＬＥＡＲ

説明 現在メモリ上にあるプログラムを残したまま、すべての数値変数を０に、文字変数を"　"（ヌルストリング）にし、配列変数は消去される。

また、ＤＥＦ文で定義した情報はすべて無効となり、現在オープンの状態になっているファイルはすべてクローズされる。

（例）
```
10 A=5:B=8:C$="MSX"
20 PRINT A;B;C$
30 CLEAR
40 PRINT A;B;C$
50 END
```

```
 5      8     MSX
 0      0
```

変数Ａ，Ｂ，Ｃ＄の値を20行で表示させると、10行で代入した値と同じ値が表示されるが、40行で表示させたときは０またはヌルストリングになる。

これは30行のＣＬＥＡＲ命令で変数が初期化されたためである。

●ＣＬＥＡＲの他の働き

ＣＬＥＡＲは変数を初期化させるほかに、メモリ領域の大きさを設定することができる。

書式 ＣＬＥＡＲ　文字領域の大きさ，メモリの上限

説明 文字領域を指定すると、文字変数を格納しておくメモリの大きさを変えることができる。

ＢＡＳＩＣの起動時は文字領域の大きさが200バイトに設定されている。文

字変数を多用すると、文字領域が不足し、Out of string spaceのエラーとなり、文字領域の大きさを拡大する必要がある。このようなときに再設定をする。

メモリの上限は、主に機械語サブルーチンを使用するときに設定する。これは、機械語プログラムがＢＡＳＩＣによって破壊されることを防ぐために、ＢＡＳＩＣが使用できるメモリの上限を指定する必要があるからである。

指定できる値は、16ＫＲＡＭであれば＆ＨＣ３１Ｆ～Ｆ３８０、また32ＫＲＡＭの場合は＆Ｈ８３１Ｆ～＆ＨＦ３８０の範囲内で指定する。

文字領域の大きさやメモリの上限が確認できないときは、Illegal function callエラーとなる。

書式例　ＣＬＥＡＲ　400，＄ＨＦ３００

文字領域400 バイトを確保し、ＢＡＳＩＣの使用できるメモリの上限を＄ＨＦ３００番地までとする。

（例）　キーボードから直接、Ａ＄＝ＳＴＲＩＮＧ＄（250，"Ａ"）と入力する。すると、Out of string spaceエラーになる。次にＣＬＥＡＲ　400 と入力した後、もう１度入力すると今度はエラーにはならない。

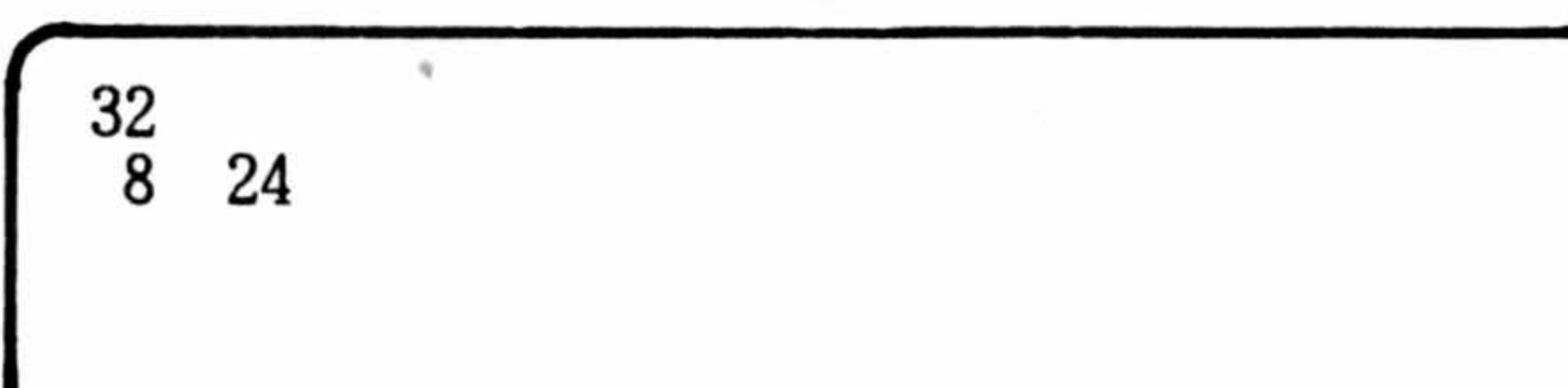
最初のエラーが出るのは、文字領域は最初 200バイトに設定されているため文字領域が足らなくなるためである。そのため、まずＣＬＥＡＲ　400で文字領域を拡大してから再入力する。

注

1　文字領域やメモリの上限を指定したときも、変数は初期化される。　配列変数だけを消去するときは、ＥＲＡＳＥ命令を使用する。

ＳＴＲ＄（エスティーアールダラー）

■数値を文字列に変換する

書式　ＳＴＲ＄（式）

説明　式の値を文字列に変換する。式は数値でも変数でもかまわない。

書式例　ＳＴＲ＄(Ａ)

変数Aの値を文字列に変換する。

```
(例)  10 A=8:B=24
      20 A$=STR$(A):B$=STR$(B)
      30 PRINT A+B
      40 PRINT A$+B$
      50 END
```

```
32
 8  24
```

10行のAとBはそれぞれ数値であるが、20行でAとBは文字列に変換されている。その結果、30行のPRINT文は、数値としてのAとBを加えた値を表示するので、32となるが、40行のPRINT文では、文字列に換えられているため数としての合計はできず、"8 24" という文字列が表示される。

注

1 数値の前には、+または−の記号を表示する場所があるが、+の場合は省略され、そのかわりにスペースが表示される。そのため文字列に変換されてもスペースが1個分文字列として残るため、文字列の前に1字分のスペースがあく。

STR$とVALは互いに逆の機能を持っている。

STRING$(ストリングダラー)

■指定した文字を指定した数だけ与える

指定した文字だけで作られた任意の長さの文字列を与える。

書式 STRING$ (文字数, 文字式) .キャラクタコード

説明 指定した文字式またはキャラクタコードに対応する文字を、指定した文字数の個数だけ並べたものを関数の値とする。

文字数が0の時は、ヌルストリング(空白)になる。文字式は文字列でも文字変数でもかまわない。

文字式が2文字以上のときは、最初の文字が使われる。

書式例　ＳＴＲＩＮＧ＄（10，ＣＨＲ＄（65））

キャラクタコードの65 "Ａ" の文字を10個並べたものを関数の値とする。

（例）
```
10 INPUT "モジハ" ;M$
20 INPUT "モジスウハ" ;K
30 A$=STRING$ (K, M$)
40 PRINT A$
50 GOTO 10
```

```
モジハ?  *
モジスウハ?  10
**********
```

まず、モジハ？と聞いてくるので、自分の好きな文字を入力する。次に、モジスウを聞いてくるので、好きな数を入力する。画面には、指定した文字が指定した文字数だけ並べて表示される。

注

1　文字数とキャラクタコードは、0〜255 の範囲で指定する。

ＳＰＡＣＥ＄（スペースダラー）

空白の文字列を作る

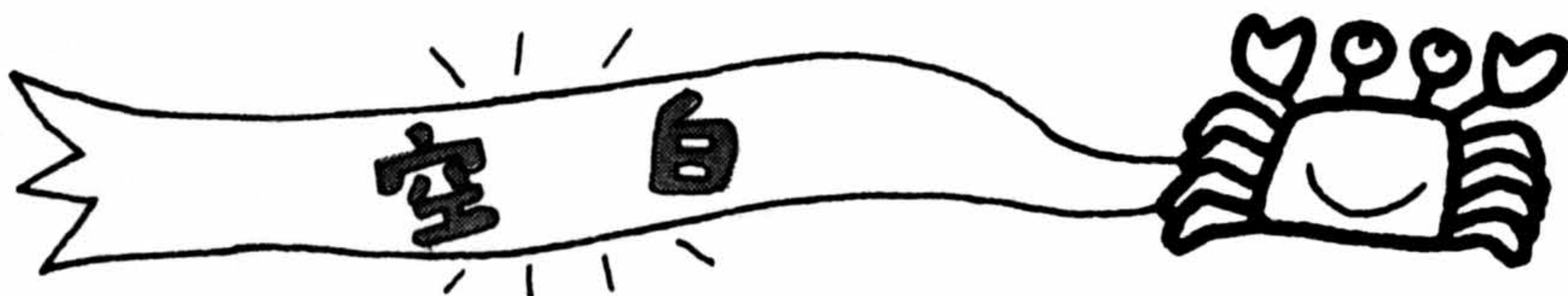
書式　ＳＰＡＣＥ＄（式）

説明　式で指定した数だけの空白（スペース）をもった文字列を与える。

式は定数、変数のどちらでもよいが、0〜255 の範囲で指定しなければならない。

書式例　ＳＰＡＣＥ＄（10）

10個の空白をもった文字列を与える。

空白

```
（例）  10  A$＝SPACE$（5）；"MSX"
        20  PRINT  A$
        30  END
```

```
          MSX
```

A$には、5個のスペースのMSXという文字が加えられたものが代入されており、PRINT文でA$を表示すると文字の前に5個分のスペースがあく。

INSTR（インストリング）

■指定した文字の位置を示す

文字列の中から指定した文字列をさがして、その位置を関数の値とする。

書式 INSTR （数式，文字列1，文字列2）

説明 文字列1の中から文字列2をさがして、見つけた位置を関数の値とする。見つからなければ、0になる。

数式は、文字列をさがし始める位置を指定するが、省略したときは文字列1の最初の文字からさがし始める。

文字列は変数で指定してもよい。

文字列2の長さが文字列1より大きい場合、また文字列1がヌルストリングのときは関数の値は0になる。

書式例 INSTR（3，A$，B$）

文字変数A$の3番目の文字からB$と同じ文字列をさがしだす。

```
（例）  10  X$＝"ABCDEFGH"
        20  S＝INSTR（X$，"FG"）
        30  PRINT  "FGハ"；S；"モジX"
        40  END
```

```
FGハ　6モジX
```

20行は、文字変数X＄の中から“FG”という文字列をさがし出して、その位置を変数Aに代入する。

注

1　数式は 0〜255 の範囲で指定する。

ABS（アブソルート）

■絶対値を得る

指定した数値の絶対値を得て、関数の値とする。

書式　ABS（式）

説明　式の値の絶対値を求める。

式は変数でも定数でもよい。また、式の値の型は整数、単精度、倍精度のいずれでもよいが、得られる値は倍精度型になる。

書式例　ABS（－2）

－2の絶対値を求める。値は2。

（例）
```
10  A=2:B=-2
20  PRINT  ABS(A)
30  PRINT  ABS(B)
40  END
```

```
2
2
```

変数AとBには、それぞれ2と－2が代入されているが、AとBの絶対値はともに2となる。

VAL（バリュー）

■文字列を数値に変換する

文字として扱われている数字を、数値として扱えるように変換する。

書式　VAL（文字式）

説明　文字式で表わす文字列を数値に変換して関数の値とする。

文字式は文字列でも文字変数でもよい。ただし、文字列の最初の文字が＋，－，＆，・，数字以外のものの場合は、ＶＡＬの値は０になる。

文字列の途中に数字以外の文字がある場合は、それ以降の文字は無視され、また、文字列の途中の空白は読みとばされる。

文字列は16進数、８進数、２進数のいずれで表示しても数値に変換できるが、数値はすべて10進数になる。

書式例　ＶＡＬ（"２７３４"）

文字列"２７３４"を数値に直す。

（例）

```
10 INPUT A$
20 A=VAL (A$)
30 PRINT A
40 GOTO 10
```

```
? 278
 278
? &HFF
 255
```

これは、ＩＮＰＵＴ文で入力した文字列を数値に変換するプログラムである。

注

1　ＶＡＬとＳＴＲ＄は互いに逆の機能を持っている。

ＬＥＮ（レングス）

■文字列の文字数をかぞえる

文字列の総文字数をかぞえて、その数を関数の値とする。

書式　ＬＥＮ（文字式）

説明　文字式の文字数をかぞえて、その数を関数の値として与える。

文字式は、文字列でも文字変数でもよい。

文字式がヌルストリング（空文字列）のときは０になる。

書式例　ＬＥＮ（"ＡＢＣＤ"）

文字列"ＡＢＣＤ"の文字数をかぞえて関数の値とする。値は４。

（例）
```
10 INPUT H$
20 S=LEN (H$)
30 PRINT S
40 GOTO 10
```

```
? TOKYO
 5
? MSX
 3
```

これは、10行のＩＮＰＵＴ文で入力した文字列の文字数を20行のＬＥＮでかぞえて、その数を画面に表示するというプログラムである。

注

1　文字列あるいは文字変数の中に、スペースまたはキャラクタコード 1～31のコントロールコードが含まれているときは画面に表示されないが、１文字としてかぞえる。

CSRLIN（カーソルライン）
POS（ポジション）

■カーソルの位置を調べる

カーソルのＸ軸の値およびＹ軸の値を調べる。

書式　CSRLIN

POS（0）

説明　ＣＳＲＬＩＮは、現在のカーソルの垂直位置、すなわちＹ軸の値を得る。値の範囲は、 0～23までである。

ＰＯＳ（0）は、現在のカーソルの水平位置、すなわちＸ軸の値を得る。

値の範囲は、ＳＣＲＥＥＮ　0の状態では 0～39、ＳＣＲＥＥＮ　1では 0～31となる。カッコの中の0はダミーなので他の定数や変数でもよいが、一般的には0を指定する。

（例）
```
10 LOCATE 8, 20
20 X=POS (0) :Y=CSRLIN
30 LOCATE 0, 20
40 PRINT X, Y
```

```
8    20
```

20行で、カーソルがどこにあるかを調べている。この場合は、10行のＬＯＣＡＴＥで指定した場所にカーソルがあるので、その位置はＸ軸が８、Ｙ軸が20ということになる。

注

１　この２つの命令はテキストモードのときに限り使用できる。

ＴＩＭＥ（タイム）

■時計機能の値を与える

1／60秒ごとに値が１ずつ増えていくシステム変数である。

書式　ＴＩＭＥ

説明　ＴＩＭＥの値は、電源を入れたとき、またはリセットボタンを押したときは０に設定され、1／60秒ごとに１ずつ増加する。

ＴＩＭＥの持つ値は 0～65535 までで、普通の変数と同じように、この範囲内の値ならばＴＩＭＥに代入することもできる。

プログラムでは、主にストップウォッチのような役目をはたすが、時計ではないので何分何秒というような表示はしない。

ＴＩＭＥの値を時刻に換算するときは次のようにする。

時　ＩＮＴ（ＴＩＭＥ／60＾3）

分　ＩＮＴ（ＴＩＭＥ／60＾2）

秒　ＩＮＴ（ＴＩＭＥ／60）

（例）
```
10 TIME=0
20 1F TIME>=60 THEN BEEP:GOTO 10
30 GOTO 20
```

これは、１秒に１回ＢＥＥＰ音を鳴らすプログラムで、20行では、ＴＩＭＥの値が60以上になったらＢＥＥＰ音を鳴らすようになっている。

ＴＩＭＥの値が60ということは、１秒たったことを意味する。

注

１　カセットテープの作動中や割り込み処理を禁止しているときは、ＴＩＭＥの値は更新されない。

ＩＮＴ（インテジャー）

■数値を超えない最大の整数を得る

書式　ＩＮＴ（式）

説明　式の値を超えない最大の整数を、関数の値とする。

式は、数値でも変数でも数式でもよい。

書式例　ＩＮＴ（Ａ）

変数Ａに代入されている数値を超えない最大の整数を関数の値とする。

（例）

```
10 INPUT K
20 PRINT INT(K)
30 GOTO 10
```

```
? 1.25
 1
? -2.3
-3
```

これはＩＮＰＵＴ文によって入力された数がＩＮＴ命令によってどのように変化するかをみるためのプログラムである。

注

1　ＩＮＴは小数点以下を切りすてるのではなく、数値を超えない最大の整数を得るための命令であるため、カッコの中がマイナスのときは注意すること。たとえばＩＮＴ（－1.2）は－2である。小数点以下切り捨てを行ないたいときはＦＩＸを使う。

ＦＩＸ（フィクス）

■小数点以下を切り捨てて整数に直す

与えられた数値の小数点以下を切り捨てた値を得る。

書式　ＦＩＸ（式）

説明　式の値の小数点以下を切り捨てた整数値を値として得る。

式は数値でも変数でも数式でもよい。

書式例　ＦＩＸ（2.4＋3.8）

2.4＋3.8　の結果、つまり　6.2　の小数点以下を切り捨てる。値は6。

（例）
```
10 INPUT S
20 PRINT FIX(S)
30 GOTO 10
```

```
? 2.38
 2
? -4.9
-4
```

これは、INPUT文によって入力された数が、FIX命令によってどのように変化するかを見るためのプログラムである。

注

1　FIX（X）とINT（X）の値はXが正のときは同じであるが、Xが負のときはちがってくる。

ASC（アスキー）

文字をキャラクタコードに変換する

指定した文字を、それに対応するキャラクタコードの値に変換する。

書式　ASC（文字式）

説明　文字式の文字をキャラクタコードに変えて、関数の値とする。

文字式は文字列でも文字変数でもよい。

文字式が複数の文字の場合は、最初の１文字だけを変換する。

書式例　ASC（"A"）

Aという文字のキャラクタコード、つまり65という値が得られる。

（例）
```
10 INPUT "モジハ";M$
20 PRINT ASC(M$)
30 GOTO 10
```

```
モジハ? H
 72
モジハ? あ
 145
```

これは入力した文字のキャラクタコードを下に表示するプログラムである。自分でいろいろためしてみること。

注

1　文字式がヌルストリングのときは、"Illegal function call"エラーに

なる。

ＡＳＣはＣＨＲ＄の逆の関数である。

ＳＴＲＩＧ（スティックトリガ）

■ジョイスティックのトリガボタンが押されたか

ジョイスティックのトリガボタンが押されたかどうかを調べ、それに応じた値を関数の値とする。

書式　ＳＴＲＩＧ（式）

説明　式の値は０～５で指定し、この値によって次のジョイスティックのトリガボタンについて調べる。

０　…………キーボードのスペースキー

１，３…………ポート１のジョイスティックのトリガボタンが押されたかどうかを調べる。

２，４…………ポート２のジョイスティックのトリガボタンが押されたかどうかを調べる。

ジョイスティックには普通２個のトリガボタンがあるため、１つのジョイスティックに２つの数値が指定できる。

ＳＴＲＩＧ関数は、指定したジョイスティックのトリガボタンが押されていれば－１、押されていなければ０の値を与える。

書式例　ＳＴＲＩＧ（０）

キーボードのスペースキーが押されているかを調べる。

（例）
```
10　T＝STRIG（0）
20　IF　T＝－1　THEN　BEEP
30　GOTO　10
```

キーボードのスペースキーが押されると、ピッというＢＥＥＰ音が鳴る。

カッコの中の値を変えてジョイスティックでもためしてみるとよい。

注

１　２つのトリガボタンを区別できないものや、ボタンが１つしかないジョイスティックの場合は、式の値に３、４を指定できないことがある。

ＳＴＩＣＫ（スティック）

■ジョイスティックの押されている方向を調べる

ジョイスティックまたはカーソルキーがどの方向に押されているかを調べる。

書式　ＳＴＩＣＫ（式）

説明　式の値は０～２で指定し、この値によって次のジョイスティックの方向を調べる。

０……キーボードのカーソルキーがどの方向に押されているかを調べる。

１……ポート１のジョイスティックがどの方向に押されているかを調べる。

２……ポート２のジョイスティックがどの方向に押されているかを調べる。

ＳＴＩＣＫ関数によって得られる値は、押された方向によって０～８の値を得る。

ジョイスティック

↑………１　↗………２　→………３　↘………４　↓………５
↙………６　←………７　↖………８　何も押されていない……０

カーソルキー

↑………１　↑＋→………２　→………３　→＋↓………４
↓………５　↓＋←………６　←………７　←＋↑………８
何も押されていない………０

（↑＋→は、カーソルキーの↑キーと→キーが同時に押されたことを意味する。）

書式例　ＳＴＩＣＫ（１）

ポート１に接続されているジョイスティックの方向を調べ、その方向に対応する値を関数の値とする。

次の例は、中央に表示されているハートマークが、カーソルキーの方向（上下左右）に動く。

ゲームなどで、キャラクタやスプライトなどを移動させるのには、このようにＳＴＩＣＫ関数の値を調べて、その値によって座標の値を加減して別の座標に表示させるという方法がよく使われる。

ただし画面内で移動する。

（例）
```
10 SCREEN 1
20 X=15:Y=12
30 LOCATE X,Y:PRINT"♥"
40 LOCATE X,Y:PRINT"  "
50 S=STICK(0)
60 IF S=1 THEN Y=Y-1
70 IF S=3 THEN X=X+1
80 IF S=5 THEN Y=Y+1
90 IF S=7 THEN X=X-1
100 GOTO 30
```

↑
←♥→
↓

PLAY（プレイ）

■音楽を演奏中かどうかを調べる

書式 PLAY（式）

説明 式は0～3までの範囲の数値で指定する。

0：すべてのチャンネルが音楽を演奏しているかどうかを調べる。

1：チャンネル1が音楽を演奏しているかどうかを調べる。

2：チャンネル2が音楽を演奏しているかどうかを調べる。

3：チャンネル3が音楽を演奏しているかどうかを調べる。

指定したチャンネルが演奏中であれば－1、そうでなければ0の値が得られる。

式の値は変数、定数、数式のいずれで指定してもよい。

（例）
```
10 CLS
20 PRINT "*";
30 IF PLAY(0)=0 THEN GOSUB 50
40 GOTO 20
50 PLAY "O4L4CDEFGABO5C"
60 RETURN
```

30行で音楽の演奏中かどうかを調べ、演奏していなければ50行からのサブルーチンを実行する。

㊅ 組み込み関数

ABS（絶対値）　（基本文型）ABS（数式）　倍精度の絶対値を得る。

SGN（符号）　（基本文型）SGN（数式）　数値の符号を調べる。数式が正から1、0なら0、負なら－1。

SQR（平方根）　（基本文型）SQR（数式）　数式が0以上のとき平方根を求める。

SIN（3角関数）　（基本文型）SIN（数式）　数式の単位はラジアンで正弦（Sin）を求める。

COS（3角関数）　（基本文型）COS（数式）　数式の単位はラジアンで余弦（Cos）を求める。

TAN（3角関数）　（基本文型）TAN（数式）　数式の単位はラジアンで正接（Tan）を求める。

ATN（逆3角関数）（基本文型）ATN（数式）　$-\frac{1}{2}\pi \sim \frac{1}{2}\pi$ の範囲で逆正接（アークタンジェント）を得る。単位はラジアン。

LOG（自然対数）　（基本文型）LOG（数式）0より大きい数式で自然対数を求める。

EXP（指数関数）　（基本文型）EXP（数式）　数式の値は145.06286085862 まで。指数関数の値を求める。

FIX（整数）　（基本文型）FIX（数式）　数式の値の小数点以下を除去した整数値を求める。

INT（整数）　（基本文型）INT（数式）　数式の値を超えない最大の整数を求める。

RND（乱数）　（基本文型）RND（数式）　乱数を求める。

CDBL（倍精度実数）　（基本文型）CDBL（数式）　整数値、単精度実数値を倍精度実数値に変換する。

CINT（整数値）　（基本文型）CINT（数式）　単精度実数値、倍精度実数値を整数値に変換する。

CSNG（単精度実数）　（基本文型）CSNG（数式）　整数値、倍精度実数値と単精度実数値に変換する。

ＰＤＬ関数（回転角読み取り関数）　（基本文型）ＰＤＬ（<パドル番号>）

パドル番号は１～12までの整数である。パドルはジョイスティックと同じポートに接続して使用するもので、ボリュームの回転角を入力する装置である。ポート１が奇数、ポート２が偶数となる。

ＰＡＤ関数（座標読み取り関数）　（基本文型）ＰＡＤ（<装置番号>）

スクリーン上の座標を読み取る、または前に読み取った点からの移動距離を読み取る。タブレット、ライトペン、マウス、トラックボールなどの装置からの座標読み込み関数である。

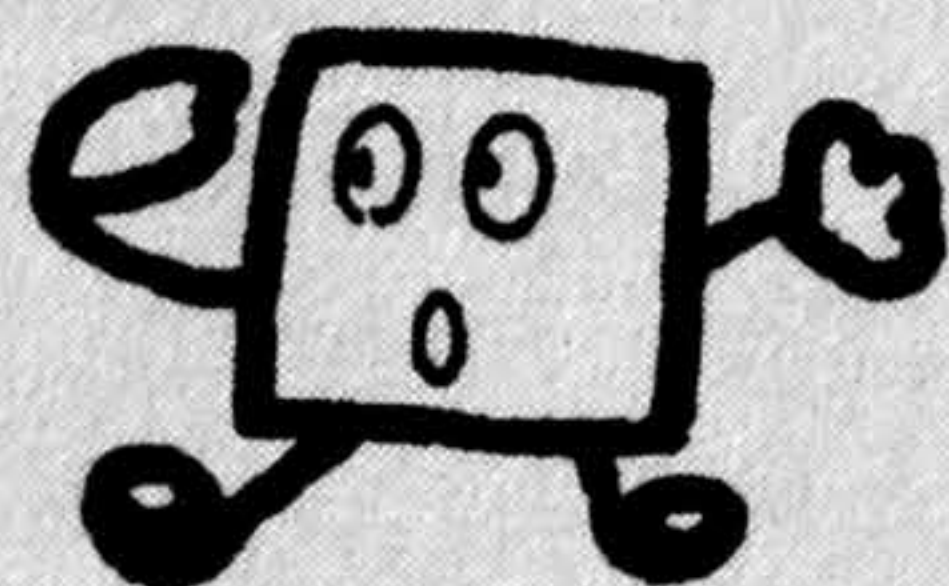
ＰＡＤ関数の装置記号と機能

装置記号	装置	返される値
0	ポート　1	ペンが押されている－1、押されていない0
1		X座標
2		Y座標
3		スイッチが押されている－1、押されていない0
4	ポート　2	ペンが押されている－1、押されていない0
5		X座標
6		Y座標
7		スイッチが押されている－1、押されていない0
8	ライトペン	データが有効－1、データは無効0
9		X座標
10		Y座標
11		スイッチが押されている－1、押されていない0
12	ポート１のマウスおよびトラックボール	入力があれば常に－１を返す
13		X座標
14		Y座標
15		入力がないときは常に０を返す
16	ポート２のマウスおよびトラックボール	入力があれば常に－１を返す
17		X座標
18		Y座標
19		入力がないときは常に０を返す

㊎ 初歩のゲームプログラム

次のプログラムは、ＭＳＸ・ＢＡＳＩＣの中のもっとも初歩的な命令だけを使った、簡単なゲームプログラムである。どのキーでも押すと、サイコロがころがる。その目の数だけπマークが移動して、標的のＸマークに命中させるというゲームだが、ひとつひとつのＢＡＳＩＣの意味や飛び先などを確認しながら、プログラム全体の流れを把握してみよう。

```
10 CLS
20 SCREEN 0,0
30 WIDTH 40:KEY OFF
40 FOR Y=1 TO 3
50 LOCATE 0,Y
60 FOR X=1 TO 40
70 PRINT"♥";
80 NEXT X,Y
90 PRINT"┌──────────────────────";
100 PRINT"────────────────────────┐"
110 FOR N=4 TO 17
120 PRINT"|
130 NEXT
140 PRINT"└──────────────────────";
150 PRINT"────────────────────────┘"
160 X1=6:Y1=6
170 LOCATE 20,11:PRINT"x"
180 LOCATE X1,Y1:PRINT"π"
190 S=0:LOCATE 12,2
200 PRINT"SAIKORO KOROKORO"
210 K$=INKEY$:IF K$=""THEN 210
220 A=INT(RND(-TIME)*6)+1
230 BEEP:S=S+1
240  IF K$=" " THEN PRINT"GOOD BY!":END
250 X=30:Y=19
260 ON A GOSUB 480,540,600,660,720,780
270 IF S=15 THEN 290
280 GOTO 220
290 C=INT(RND(-TIME)*6)+1
300 LOCATE X1,Y1:PRINT" "
310 IF (C=2)OR(C=3) THEN X1=X1+A
320 IF C=5 THEN X1=X1-A
330 IF C=6 THEN Y1=Y1-A
340 IF (C=1)OR(C=4) THEN Y1=Y1+A
350 IF X1<=2 THEN X1=2
360 IF X1>=37 THEN X1=37
```

```
370 IF Y1<=5 THEN Y1=5
380 IF Y1>=18 THEN Y1=18
390 LOCATE X1,Y1:PRINT"π"
400 IF (X1=20)AND(Y1=11) THEN 420
410 S=0:GOTO 210
420 LOCATE 19,10:PRINT"END"
430 FOR M=1 TO 20
440 BEEP:LOCATE 19,10:PRINT"   "
450 FOR T=1 TO 100:NEXT
460 LOCATE 19,10:PRINT"END"
470 NEXT:END
480 LOCATE X,Y:  PRINT"┌───┐"
490 LOCATE X,Y+1:PRINT"│   │"
500 LOCATE X,Y+2:PRINT"│ ● │"
510 LOCATE X,Y+3:PRINT"│   │"
520 LOCATE X,Y+4:PRINT"└───┘";
530 RETURN
540 LOCATE X,Y:  PRINT"┌───┐"
550 LOCATE X,Y+1:PRINT"│o  │"
560 LOCATE X,Y+2:PRINT"│   │"
570 LOCATE X,Y+3:PRINT"│  o│"
580 LOCATE X,Y+4:PRINT"└───┘";
590 RETURN
600 LOCATE X,Y:  PRINT"┌───┐"
610 LOCATE X,Y+1:PRINT"│o  │"
620 LOCATE X,Y+2:PRINT"│ o │"
630 LOCATE X,Y+3:PRINT"│  o│"
640 LOCATE X,Y+4:PRINT"└───┘";
650 RETURN
660 LOCATE X,Y:  PRINT"┌───┐"
670 LOCATE X,Y+1:PRINT"│o o│"
680 LOCATE X,Y+2:PRINT"│   │"
690 LOCATE X,Y+3:PRINT"│o o│"
700 LOCATE X,Y+4:PRINT"└───┘";
710 RETURN
720 LOCATE X,Y:  PRINT"┌───┐"
730 LOCATE X,Y+1:PRINT"│o o│"
740 LOCATE X,Y+2:PRINT"│ o │"
750 LOCATE X,Y+3:PRINT"│o o│"
760 LOCATE X,Y+4:PRINT"└───┘";
770 RETURN
780 LOCATE X,Y:  PRINT"┌───┐"
790 LOCATE X,Y+1:PRINT"│ooo│"
800 LOCATE X,Y+2:PRINT"│   │"
810 LOCATE X,Y+3:PRINT"│ooo│"
820 LOCATE X,Y+4:PRINT"└───┘";
830 RETURN
```

⑧ディスクコマンド

SAVE（セーブ）

■ディスクにファイルをセーブする

ＣＳＡＶＥはカセットにプログラムをセーブする命令だが、ＳＡＶＥはディスクにファイルをセーブする。

書式 ＳＡＶＥ　”〈(デバイス名)：(ファイル名)〉(，Ａ)”

説明　ファイルをセーブする命令である。(デバイス名）は、接続された各周辺装置の名称で、大文字で指定しても小文字で指定してもかまわない。

うしろの「，Ａ」を省略した場合は、バイナリ形式のセーブになり、Ａをつけることによってアスキー形式でセーブされる。ＭＥＲＧＥ（マージ）命令でプログラムをつなげる場合には、アスキー形式でセーブされていなければならない。ＲＡＭディスクのセーブはすべてアスキー形式になる。

各デバイス名には次のものがある。

ＣＡＳ：………………カセット――――――――入力と出力がある
Ａ：～Ｈ：まで………フロッピーディスク――入力と出力がある
ＭＥＭ：………………ＲＡＭディスク――――入力と出力がある
ＧＲＰ：………………グラフィック画面―――出力モードのみ
ＣＲＴ：………………文字画面――――――――出力モードのみ
ＬＰＴ：………………プリンタ――――――――出力モードのみ

ＳＡＶＥ命令では、ＣＲＴやＬＰＴにもプログラムを出力することができるが、このデバイスを指定した場合、それぞれＬＩＳＴ命令、ＬＬＩＳＴ命令を実行したのと同じ状態になる。

カセットにセーブした場合には、ボーレートの指定ができる。

ＣＳＡＶＥ”(ファイル名)，(ボーレート)”

ボーレートは、１を指定すると1200ボー、２を指定すると2400ボーで、それぞれセーブされる。

LOAD（ロード）

■ディスクからファイルを読み出す

書式 LOAD ”〈(デバイス名)：(ファイル名)〉(，R)”

説明 ディスクからファイルを読み出す命令で、使用デバイスは、ディスク、RAMディスク、カセットなどである。

Rをつけると、ファイルを読み出すとすぐに実行を開始する。

ファイル形式（バイナリ形式かアスキー形式）は、ディスク、RAMディスクの場合には、自動的に判別されて読み出されるが、カセットの場合には判別しないので注意する。

書式例 バイナリ形式…………CLOAD（ディスク名）
　　　　アスキー形式…………LOAD ”〈(CAS：)(ファイル名)(，R)”

ボーレートは、自動的に判別して読み出される。

LOAD ”A：TEST．BAS”

ＢＳＡＶＥ（ビーセーブ）
ＢＬＯＡＤ（ビーロード）

■二進数や機械語のセーブ、ロード

ＢＳＡＶＥは、指定した範囲のメインメモリまたはビデオＲＡＭの内容を二進数でセーブする。ＢＬＯＡＤは、機械語プログラムをロードする。

書式

ＢＳＡＶＥ　”デバイス名：ファイル名．タイプ名”，先頭番地，終了番地，実行開始番地，（ＲまたはＳ） ＢＬＯＡＤ　”デバイス名：ファイル名．タイプ名”，（ＲまたはＳ），オフセット

（注）Ｒはロード後実行、Ｓはビデオメモリ。

書式例　ＢＳＡＶＥ　”Ａ：ＡＢＣ．ＪＯＢ”，＆h0000，＆h2900
ＢＬＯＡＤ　”Ａ：ＡＢＣ．ＪＯＢ”，Ｒ

ＦＩＬＥＳ（ファイルス）

■ディスク内のファイル名を表示させる

ディスク内にセーブしてあるプログラムのファイル名をすべて表示する。

書式　ＦＩＬＥＳ

ＦＩＬＥＳ（ファイルス）
………ディスク内のファイルをすべて画面に表示する。

ＬＦＩＬＥＳ（エルファイルス）
………ディスク内のファイルすべてをプリントアウトする。

ＣＡＬＬ　ＭＦＩＬＥＳ（コール　エムファイルス）
………ＲＡＭディスク内のファイルをすべて画面に表示する。
「ＣＡＬＬ」は”_ＭＦＩＬＥＳ”のように「＿」（アンダーバー）でも代用できる。

いずれも該当するファイルが見つからなかったときには、「File not found」

エラーが表示される。

書式例　FILES“A：TEST1. BAS”

デバイス名とファイル名は必ず「”」（ダブルクォーテーション）で囲む。

プリンタへの出力命令としては、PRINT命令に相当するLPRINT命令と、FILESに相当するLFILES、LISTに相当するLLIST命令を使用することができる。

RUN((ラン)

■プログラムを呼び出して実行する

LOAD命令とRUN命令を兼用し、ディスクにあるプログラムを呼び出すと同時に実行させる。

書式　RUN”デバイス名：ファイル名. タイプ名”, R

書式例　RUN” A：TEST1. ASC”, R

……ファイル名TEST1、タイプ名ASCのプログラムを呼び出して実行する。

このときファイルはすべてクローズする。Rをつけることによって、ファイルをクローズしないでプログラムをスタートさせる。

注　RUNはプログラムを実行するが、プログラムをロードするだけの命令がLOADである。

書式　LOAD”デバイス名：ファイル名. タイプ名”, R

●**AUTOEXEC. BAS**（オートスタートプログラムファイル）

電源をONして、DISK　BASICをスタートさせたとき、プログラムを自動的にロードして実行させるなら、このファイル名で、プログラムをセーブしておく。

ＫＩＬＬ／ＭＫＩＬＬ（キル／エムキル）

■ディスク内のファイルを消去する

ディスク内にあるプログラムファイルのうち、不要になったファイルを消去する。

書式 ＫＩＬＬ”デバイス名：ファイル名．タイプ名”

書式例 ＫＩＬＬ”Ａ：ＴＥＳＴ1．ＢＡＳ”……ファイル名ＴＥＳＴ1のファイルを消去する。

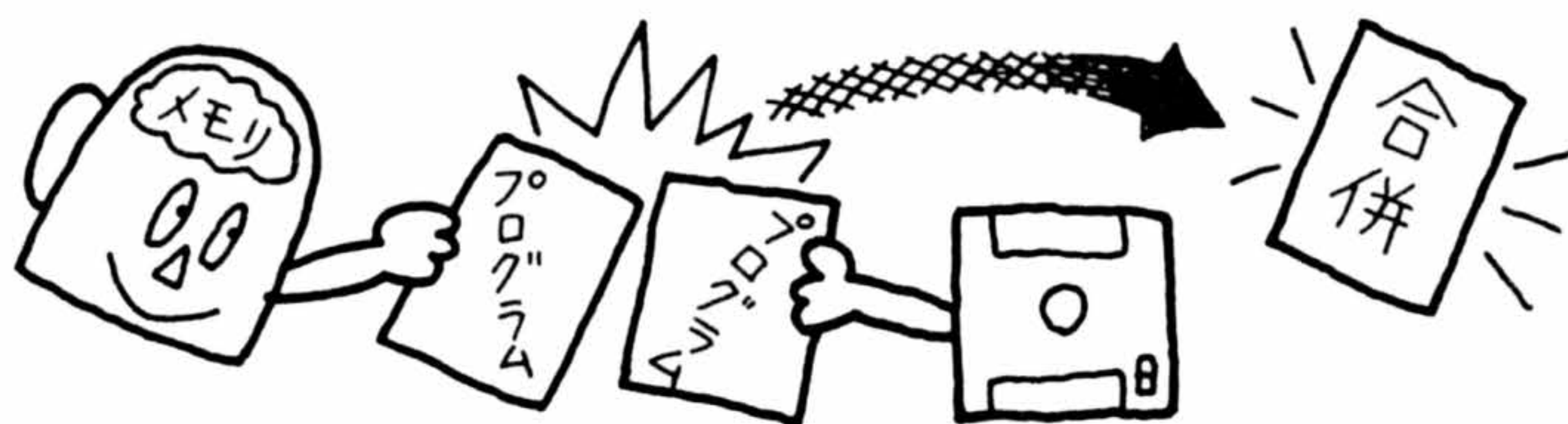
注 ＭＫＩＬＬは、ＲＡＭディスクのファイルのみを対象とする。
書式はＫＩＬＬ命令に準ずる。

ＭＥＲＧＥ（マージ）

■二つのプログラムを合併する

メモリにあるプログラムとディスクにあるプログラムを合併する。二つのプログラムを合併して一つのプログラムにする命令で、アスキー形式でセーブされたプログラムだけに適用される。

書式 ＭＥＲＧＥ”デバイス名：ファイル名．タイプ名”

書式例 ＭＥＲＧＥ”Ａ：ＴＥＳＴ2．ＡＳＣ”……ディスクにあるファイル名ＴＥＳＴ2、タイプ名ＡＳＣのプログラムをメモリにあるプログラムに合併する。

⑨ファイル制御

OPEN（オープン）

■データファイルのアクセスを開始する

指定するデバイスにデータファイルのアクセスを開始する命令である。ファイルへのアクセス方法には、シーケンシャルファイルの場合と、ランダムアクセスファイルの場合とがあり、どちらもこのＯＰＥＮ命令で開くことができる。

書式

・シーケンシャルファイルの場合
　ＯＰＥＮ（”〈デバイス名〉：〈ファイルネーム〉”）ＦＯＲ（モード）ＡＳ ＃（ファイル番号）
・ランダムアクセスファイルの場合
　ＯＰＥＮ（”〈デバイス名〉：〈ファイルネーム〉”）ＡＳ ＃（ファイル番号）［ＬＥＮ＝〈コード長〉］

説明　モードはファイルへのアクセス方法を指定するもので、シーケンシャルファイルには次の３つのがある。

ＩＮＰＵＴ…………指定されたファイルからのデータの読み出しを行なう。
ＯＵＴＰＵＴ………指定されたファイルを作成しデータの書き込みを行なう。
ＡＰＰＥＮＤ………すでに作成されているファイルに対して、追加書き込みを行なう。

ファイル番号とは、ファイルの操作に関するワークエリアを割り当てるためのもので、以後、データをアクセスするときにはこの番号を指定する。指定できる文字数は１～15の範囲で整数値を指定する。

レコード長は、１つのデータの長さを表わしたものである。１レコードは、１～255バイトの数値で指定する。ランダムファイルは固定長だが、シーケン

シャルファイルはあとからレコードを追加することができる。１レコードの割り当ての定義は、ＦＩＥＬＤ命令で行なう。

ＯＰＥＮ命令でファイルを開き処理を実行したあとは、必ずＣＬＯＳＥ文でファイルを閉じる。このとき、ＯＰＥＮで開いたファイル番号と同じファイル番号でファイルを閉じる。

書式例

• シーケンシャルファイルの場合

ＯＰＥＮ ”Ａ：ＴＥＳＴ1. ＤＡＴ” ＦＯＲ ＩＮＰＵＴ ＡＳ ＃1

ディスクＡのファイル名”ＴＥＳＴ１. ＤＡＴ”のファイル番号１をオープンし、ファイルからの読み出しを開始する。

```
(例) 10 'シーケンシャルファイル ヨミコミ
     20 OPEN "A:TEST1. DAT" FOR INPUT AS
        #1
     30 IF EOF(1) THEN 90
     40 INPUT #1, A$
     50 INPUT #1, B$ (INPUT #1, A$, B$, C$で
        もよい)
     60 INPUT #1, C$
     70 PRINT A$;" ";B$ " ";C$
     80 GOTO 30
     90 CLOSE #1
    100 END
```

• ランダムファイルの場合

ＯＰＥＮ ”Ａ：ＴＥＳＴ2. ＤＡＴ” ＡＳ ＃2 ＬＥＮ＝20

ディスクＡのファイル名”ＴＥＳＴ2. ＤＡＴ”のファイル番号２を指定してファイルをオープンするが、指定したファイルが存在しない場合は、新しいファイルを作成する。

シーケンシャルファイルと違ってモードがないために、ランダムファイルはファイルのどこからでも読み込みが可能である。

（例）
```
10 'ランダムファイル　ヨミコミ
20 OPEN "A:TEST2.DAT" AS #2 LEN=20
30 FIELD #2, 8 AS A$, 12 AS B$
40 GET #2, 1
50 PRINT A$, B$
60 CLOSE #2
70 END
```

ＣＬＯＳＥ（クローズ）

■ファイルの操作を終了する

ＯＰＥＮ文でファイルを開き処理を実行したあと、ＣＬＯＳＥ文でファイルを閉じ、操作を終了する。

書式　CLOSE [#〈ファイル番号〉]

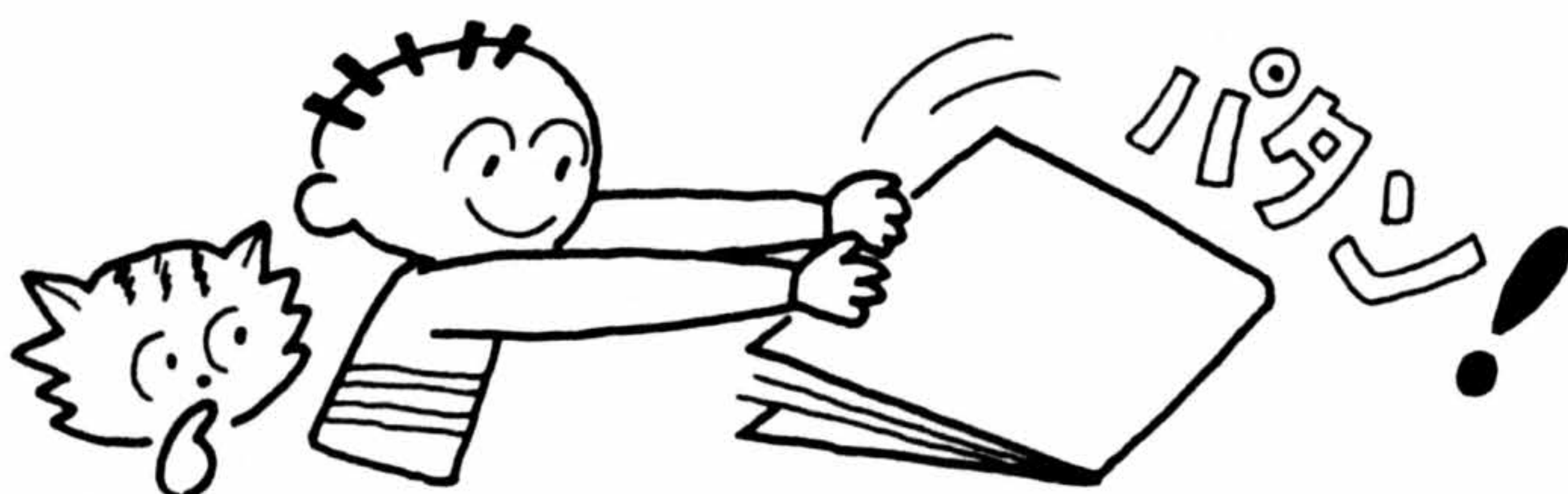
説明　指定したファイルを閉じる命令で、現在ＯＰＥＮされているファイル番号で指定する。

ファイルを開いている場合、ＥＮＤ文やＮＥＷコマンドを実行すると、ファイルを閉じることができる。

書式例　CLOSE #1, #3

ファイル番号の１と３を閉じる。

CLOSE

ＣＬＯＳＥだけを指定すると、現在開かれているファイルのすべてを閉じる。

PRINT＃（プリントシャープ）

PRINT＃USING（プリントシャープユージング）

■シーケンシャルファイルにデータを書き込む

シーケンシャルファイルにデータの書き込みを行なう。

書式

① PRINT ＃（ファイル番号），（変数）
② PRINT ＃（ファイル番号），USING 〈書式〉；（変数）

説明 ファイル番号で指定したファイルに、データを書き込む。

①の書式でのデータは、変数、文字変数のどちらでも、データとして扱う。

②の変数は、数値だけを指定できる。

PRINT ＃ USINGは、PRINT USINGを参照のこと。

INPUT＃（インプットシャープ）

LINE INPUT＃（ラインインプットシャープ）

■指定されたファイルからデータを読み込む

指定されたファイルから、変数の数だけデータを読み込む。

書式

① INPUT ＃（ファイル番号），（変数）
② LINE INPUT＃ （ファイル番号），（変数）

説明 指定されたファイルから、データを読み込む。INPUT＃はINPUT文と同じで、数字および文字型変数のどちらでも読み込みが可能だが、ファイルを書き込んだ形式と同じでなければならない。つまり、PRINT＃文で数値変数で書き込んだら、INPUT＃文で読み出すときも数値変数で読み出す。

LINE INPUT＃は、LINE INPUT文と同じで、1行単位でデータを読み込むので、「,」（カンマ）や「”」ダブルクォーテションなども無視して読み込む。

ＬＩＮＥ　ＩＮＰＵＴは、変数にキャリッジリターンがくるまでのすべての文字列をストリング変数に代入することができる。

ＥＯＦ（エンド オブ ファイル）

■データの読み込み終了を調べる

オープンされているファイルのデータを最後まで読み取ったかを調べる。

書式　ＥＯＦ（ファイル番号）

説明　結果は、データが最後まで読み込まれた場合は真（−1)、データが残っている場合は偽（１）を数値で返す。

書式例　ＩＦ　ＥＯＦ（１）ＴＨＥＮ　ＣＬＯＳＥ　#1

ファイル番号１がファイルエンドになると、調べた結果を真（−１）の値で返し、ファイル処理を終了する。

ＰＵＴ#（プットシャープ）

■ランダムファイルにデータを書き込む

書式　ＰＵＴ#（ファイル番号)，[(レコード番号)]

説明　ランダムアクセスファイルにデータを書き込む。データの書き込みは、ＦＩＥＬＤ文で指定している変数にデータを設定したあと、ＰＵＴ#文でレコード番号を指定して書き込む。

レコード番号は１以上の整数で指定し、省略すると、以前のＰＵＴ#文またはＧＥＴ#文でアクセスされた次のレコードに指定される。

また、ＦＩＥＬＤ文で定義された変数データを設定するには、ＬＳＥＴ文やＲＳＥＴ文によって、あらかじめファイルバッファに書き込まれていなければならない。

書式例　ＰＵＴ　#1，１

```
(例) 100　FIELD　#1, 10　AS　名前$, 10　AS　年$
     110　LSET　名前$＝A$
```

```
120　RSET　年$＝B$
130　PUT　#1, 1
```
（この例は、使い方の一例で、このままでは作動しないので注意する）

GET#（ゲットシャープ）

■ランダムファイルからデータを読み込む

書式　GET　#（ファイル番号），[（レコード番号）]

説明　ファイル番号で指定されているディスクファイルの、レコード番号で指定されているレコードを、FIELD文で指定されている変数にデータとして設定する。

レコード番号は1以上の整数で指定し、省略すると、以前にPUT#文またはGET#文によってアクセスされた次のレコードになる。

書式例　GET　#1, 2

（例）
```
100　FIELD　#1, 10　AS　名前$, 10　AS　年$
110　GET　#1, 1
120　PRINT　名前$；年$
```
（この例は、使い方の一例で、このままでは作動しないので注意する）

MAX FILES（マックスファイルス）

■同時に複数のファイルをオープンする

プログラムで同時に二つ以上のファイルを開くとき、その数を指定してファイルを開く。

書式　MAXFILES＝ファイル数

書式例　MAXFILES＝3………三つのファイルを開く。

open

FIELD（フィールド）

■ランダムファイルのレコード長を設定する

書式

```
FIELD #（ファイル番号），（フィールド長） AS （文字変数）
 [, ………………]
```

説明　オープンされたファイルの、1レコード内のデータの割り当てを定義する命令で、そのデータのバイト数で定義する。

レコード長は1～255の数値で指定し、それが1レコードのバイト数になる。

OPEN文で指定されたレコード長の範囲内であれば、いくつでも設定できる。

書式例　FIELD #1, 10 AS 名前$, 5 AS 年$

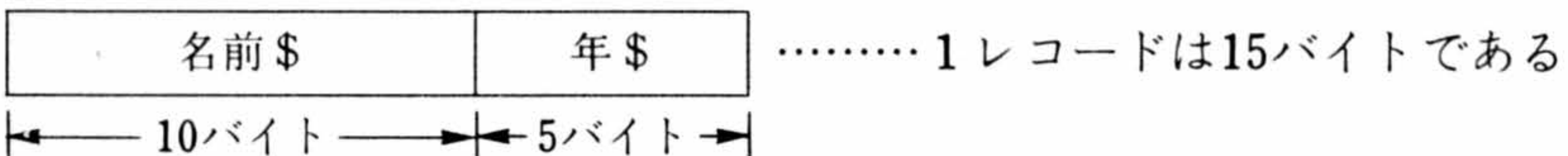
レコード長は名前$が10と年$が5の、合計15になる。OPEN文でレコード長を15と設定していれば、これが1レコード全体をアクセスすることになるが、もしOPEN文でレコード長を30と設定していると、残りの部分は未使用部分になり、ファイルをむだに使っていることになるので注意する。

OPEN文　レコード長＝15設定

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

名前$	年$

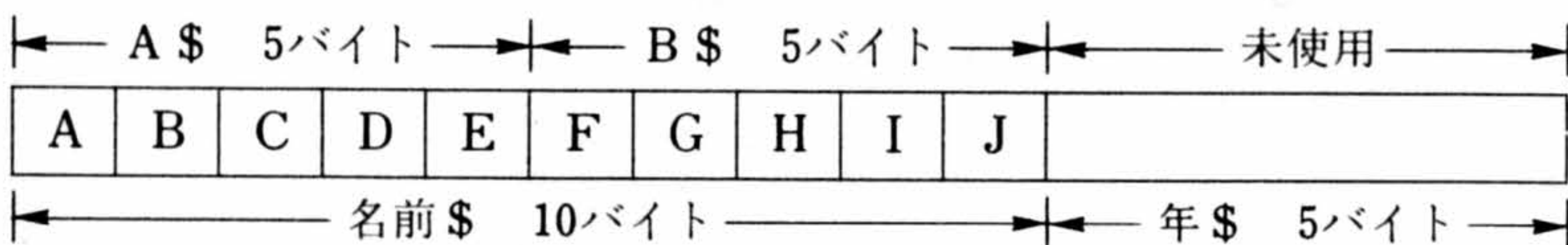
………1レコードは15バイトである

|←── 10バイト ──→|← 5バイト →|

```
(例) 100 FIELD #1, 10 AS 名前$, 5 AS 年$
     110 FIELD #1, 5 AS A$, 5 AS B$
```

このように、FIELD文による文字型変数の割り当ては、一つのファイルバッファに何回でも設定することができ、これらの定義はすべて有効となる。

|← A$ 5バイト →|← B$ 5バイト →|← 未使用 →|

A	B	C	D	E	F	G	H	I	J	

|←── 名前$ 10バイト ──→|← 年$ 5バイト →|

この設定では、A$にはABCDEが、B$にはFGHIJがそれぞれ得られることになる。

ＬＳＥＴ（レフトセット）
ＲＳＥＴ（ライトセット）

■フィールドにデータを設定する

書式

ＬＳＥＴ　（文字型変数）＝（文字列型データ）
ＲＳＥＴ　（文字型変数）＝（文字列型データ）

説明　ファイルバッファへデータを書き込もうとするとき、通常の代入文を使うと、一般の変数用の領域にデータを割り当ててしまい、ＦＩＥＬＤ文による定義が失われてしまう。そこで、ファイルバッファに直接データを書き込むＬＳＥＴ文とＲＳＥＴ文で、データを設定しておく。

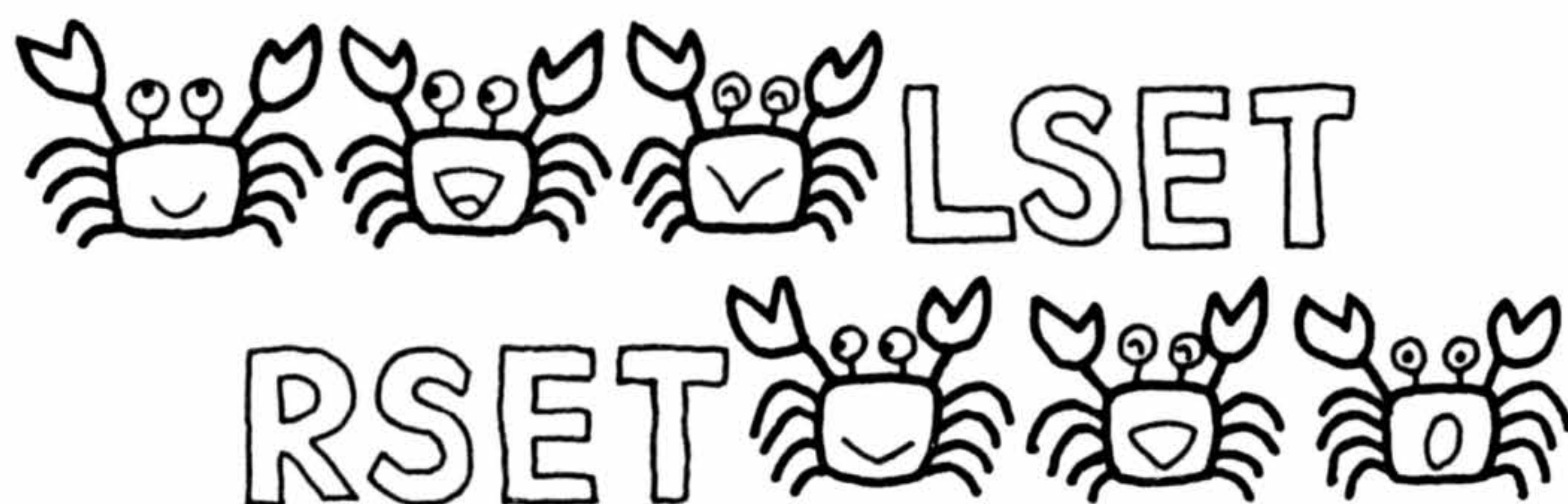
書式例　ＬＳＥＴ　Ａ＄＝“ＡＢＣＤＥ”
　　　　　ＲＳＥＴ　Ｂ＄＝“１２３４”

ＬＳＥＴ文は文字領域内に左詰めで代入し、ＲＳＥＴ文は文字列領域内に右詰めで代入する。それぞれ領域の空いた部分には空白が詰められる。

また、文字列の長さが指定された変数の領域を超えた場合には、ＬＳＥＴ文、ＲＳＥＴ文とも、文字列の長い部分の右側を切り捨てる。

（例）
```
100　ＦＩＥＬＤ　#1, 5　ＡＳ　Ａ＄, 5　ＡＳ　Ｂ＄
110　ＬＳＥＴ　Ａ＄＝“ＡＢＣＤＥ”
120　ＲＳＥＴ　Ｂ＄＝“１２３４”
```

A$やB$の変数は、ＦＩＥＬＤ文で定義されている変数でなければならない。

LSET
RSET

LOF(レングス オブ ア ファイル)

■指定されたファイルの大きさを求める関数

書式 LOF (〈ファイル番号〉)

説明 ファイル番号でオープンされているファイルの大きさをバイト数で返す。

シーケンシャルファイルの場合は使用セクタ数を返し、ランダムファイルでは、使用している最大のレコード番号を返す。

LOC(ロケーションカウンター)

■指定されたファイルの論理的な現在位置を求める

書式 LOC (〈ファイル番号〉)

説明 ファイル番号でオープンされているファイルに対して、現在どの位置をアクセスしているかを求める関数で、ファイルの先頭からの位置を返す。

シーケンシャルファイルの場合には、そのファイルがオープンされてから読み込まれた、または書き込まれたデータのバイト数を256単位で返す。ランダムファイルでは、最後に読み込まれた、または書き込まれたレコード番号を返す。

ＭＫＩ＄(メイクインテジャーダラー)

ＭＫＳ＄(メイクシングルダラー)

ＭＫＤ＄(メイクダブルダラー)

■数値データを内部表現に対応した文字列に変換

書式

```
ＭＫＩ＄　（〈整数型データ〉）
ＭＫＳ＄　（〈単精度型データ〉）
ＭＫＤ＄　（〈倍精度型データ〉）
```

説明　ランダムファイルへの書き込みは、すべて文字列で行なわなければならない。そのため、数値データをランダムファイルへ書き込む場合には、ＭＫＩ＄、ＭＫＳ＄、ＭＫＤ＄を使って文字型に変換する。

結果の文字長はＭＫＩ＄（整数型）が２文字、ＭＫＳ＄（単精度型）が４文字、ＭＫＤ＄（倍精度型）が８文字にそれぞれ文字列に変換する。これは、ＢＡＳＩＣ内部では、整数型、単精度型、倍精度型の数値が、それぞれ２バイト、４バイト、８バイトで表現されているためである。

逆に、これらのデータを読み出すには、ＣＶＩ／ＣＶＳ／ＣＶＤの関数を使用する。

書式例

```
ＦＩＥＬＤ　#1, 2　ＡＳ　Ａ＄, 4　ＡＳ　Ｂ＄, 8　ＡＳ
　Ｃ＄
110　ＬＳＥＴ　Ａ＄＝ＭＫＩ＄（Ａ）
120　ＬＳＥＴ　Ｂ＄＝ＭＫＳ＄（Ｂ）
130　ＬＳＥＴ　Ｃ＄＝ＭＫＤ＄（Ｃ）
140　ＰＵＴ　#1, 1
```

CVI(コンバートインテジャー)
CVS(コンバートシングル)
CVD(コンバートダブル)

■文字列データを数値データに変換する

書式

```
CVI (〈2文字の文字列式〉)
CVS (〈4文字の文字列式〉)
CVD (〈8文字の文字列式〉)
```

説明　内部表現に対応した文字列データを数値データに変換する関数である。

3つの関数は、それぞれMKI$、MKS$、MKD$の逆関数になっており、ランダムファイルから読み込んだ文字列型変数をもとの数値変数に戻す働きをする。

つまり、CVI／CVS／CVDと、MKI$／MKS$／MKD$の各関数は対応していて、たとえばMKI$で変換した文字列は、CVI関数で数値に戻さなければならない。

書式例

```
100 FIELD #1, 2 AS A$, 4 AS B$, 8 AS
    C$
110 GET #1, 1
120 A=CVI (A$)
130 B=CVS (B$)
140 C=CVD (C$)
```

この書式例は、「MKI$／MKS$／MKD$」の書式例と対応しているので参照のこと。

㊅ 初歩のランダムファイル・プログラム

ランダムファイルの働きを理解するための初歩的な「住所録プログラム」である。メインルーチンに〝データの検索〟〝変更〟などを加えて、実用化してみよう。85行と115行は追加プログラムで、プログラムの実行を終了させたいときにファイルをクローズする文である。

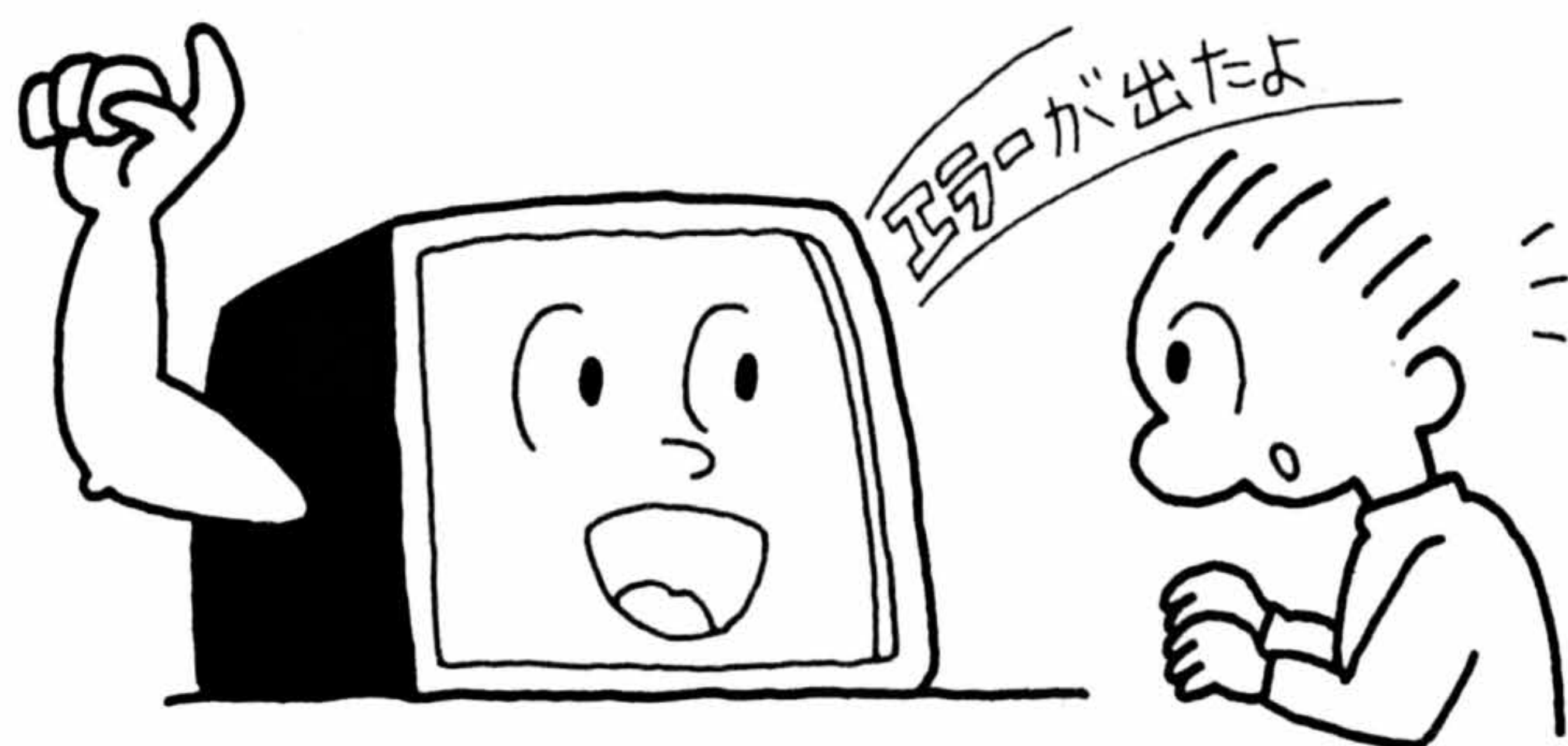
プログラム作動中にブレークをかけたときは、ファイルを壊さないために、必ずダイレクトモードでファイルを閉じる（CLOSE ＃1）。

```
10  ' ジュウショロク                                   ┐
20  DEFDBL A-Z                                         ├ 初期設定
30  INPUT "ファイルメイ=";NM$                          ┘
40  OPEN NM$ AS #1 LEN=32                              ─ ファイルオープン
50  FIELD #1, 12 AS NA$, 12 AS J$, 8 AS T$            ┘
60  ' メイン                                           ┐
70  PRINT "[1]ニュウリョク"                             │
80  PRINT "[2] データノヒョウジ"                        │
85  PRINT "[3] END"                                    │
90  B$=INKEY$                                          ├ メイン
100 IF B$="1" THEN 130                                 │
110 IF B$="2" THEN 200                                 │
115 IF B$="3" THEN 280                                 │
120 GOTO 90                                            ┘
130 'データ, ニュウリョク                               ┐
140 INPUT "ナマエハ    ";X$                            │
150 INPUT "ジュウショハ";Y$                             │
160 INPUT "デンワハ    ";Z$                            ├ データの入力
170 LSET NA$=X$:LSET J$=Y$:LSET T$=Z$                  │
180 PUT #1, LOF(1)/32+1                                │
190 GOTO 60                                            ┘
200 ' データノヒョウジ                                  ┐
210 FOR N=1 TO LOF(1)/32                               │
220 GET #1, N                                          │
230 PRINT "ナマエ    = ";NA$                           │
240 PRINT "ジュウショ= ";J$                             ├ データの表示
250 PRINT "デンワ    = ";T$                            │
260 NEXT N                                             │
270 GOTO 60                                            ┘
280 ' オワリ                                           ┐
290 CLOSE #1                                           ├ 終了
300 END                                                ┘
```

㊟ エラーメッセージ

Bad file name	ファイル名が適当でない。
Bad file number	オープンしていないファイル番号を指定した。
Can't continue	ＣＯＮＴコマンドでもプログラムを再開できない。
Device I/O error	周辺装置との入出力関係エラー。たとえば接続ミス。
Direct statement	ＬＯＡＤしているときプログラム以外のデータがあった。
Division by zero	やってはいけない０で徐算した。０に負のべき乗をした。
FIELD over flow	フィールドサイズが256以上になった。
File already open	指定したファイル番号を他のファイルで使用中。
File not found	指定したファイルが見つからない。
File not open	ＯＰＥＮされていないファイルに入出力した。
Illegal direct	ダイレクトモードではできないものをしようとした。
Illegal function call	ステートメントや関数の誤使用。引数の範囲超過。
Input past end	データを全部読んだのにまた入力してしまった。
Internal error	ＢＡＳＩＣ内部のエラー。
Missing operand	文中のパラメータやコンマ、ピリオドの使用法のまちがい。
NEXT without FOR	ＦＯＲ～ＮＥＸＴ文のＦＯＲがない。
Out of DATA	ＲＥＡＤ～ＤＡＴＡ文だが読み出すデータがない。

Out of memory	指定がメモリの容量を超えている。
Out of string space	文字列変数の文字領域が不足。
Overflow	許容範囲を超えた演算結果や数値の入力実数計算の結果が範囲を超えた場合。
Redimensioned array	配列変数を二重に定義したなど。
RETURN without GOSUB	ＧＯＳＵＢ文とＲＥＴＵＲＮ文が正しく対応していない。
Subscript out of range	配列変数の添え字が最大値より大きい。
Syntax error	文の構文や文法が誤っている。
Type mismatch	数式でなければならないときに文字列を使ったりした。
Undefined line number	指定した行番号がない。
Bad FAT	ディスケットが破壊した。
Bad drive name	ドライブ名がまちがっている。
Disk write protected	ディスクが書き込み禁止になっている。
Disk I/O error	使用中のディスクに読み込み、書き込みのエラーが発生した。
Disk offline	ディスクが使用不可能。

オールMSX
BASIC用語・用例新辞典

発　行	1990年4月20日
定　価	680円(本体660円)
著　者	秋本京子
発行者	深見兵吉
印刷所	広研印刷株式会社
発行所	成美堂出版 〒112 東京都文京区水道1-8-2 ☎(03)814-4351